

**Prepared by Rami Reddy (sun certified professional)**

**Ur valuable suggestions on this material send to  
ramireddy.satya@gmail.com**

getting success in INTERVIEW

1. good at communication & be expressive
2. enthusiasm
3. eye contact
4. be natural
5. confident & straight answer

HR round TIPS

1. Why do you want to work at our company?

Best sources for researching your target company:

annual reports, the corporate newsletter, contacts you know at the company or its suppliers, advertisements, articles about the company in the trade press.

2. What are your greatest strengths ?

Prior to any interview, you should have a list mentally prepared of your greatest strengths. You should also have, a specific example or two, which illustrates each strength, an example chosen from your most recent and most impressive achievements.

### 3. Why are you leaving (or did you leave) this position ?

(If you have a job presently tell the HR)

If you're not yet 100% committed to leaving your present post, don't be afraid to say so. Since you have a job, you are in a stronger position than someone who does not. State honestly what you'd be hoping to find in a new spot.

### 4. Why have you been out of work so long ?

You want to emphasize factors which have prolonged your job search by your own choice.

Example: "After my job was terminated, I made a conscious decision not to jump on the first opportunities to come along. In my life, I've found out that you can always turn a negative into a positive IF you try hard enough. This is what I determined to do. I decided to take whatever time I needed to think through what I do best, what I most want to do, where I'd like to do it...and then identify those companies that could offer such an opportunity."

### 5. Tell me about a situation when your work was criticized ?

Begin by emphasizing the extremely positive feedback you've gotten throughout your career and (if it's true) that your performance reviews have been uniformly excellent.

Of course, no one is perfect and you always welcome suggestions on how to improve your performance. Then, give an example of a not-too-damaging learning experience from early in your career and relate the ways this lesson has since helped you. This demonstrates that you learned from the experience and the lesson is now one of the strongest breastplates in your suit of armor.

### 6. May I contact your present employer for a reference ?

Express your concern that you'd like to keep your job search private, but that in time, it will be perfectly okay.

Example: "My present employer is not aware of my job search and, for obvious reasons; I'd prefer to keep it that way. I'd be most appreciative if we kept our discussion confidential right now. Of course, when we both agree the time is right, then by all means you should contact them. I'm very proud of my record there.

7. Give me an example of your creativity (analytical skill...managing ability, etc.)

present any of your achievements in light of the quality the interviewer is asking about. For example, the smashing success you orchestrated at last year's trade show could be used as an example of creativity, or analytical ability, or your ability to manage.

8. Why have you had so many jobs ?

First, before you even get to the interview stage, you should try to minimize your image as job hopper. If there are several entries on your resume of less than one year, consider eliminating the less important ones. Perhaps you can specify the time you spent at previous positions in rounded years not in months and years.

Example: Instead of showing three positions this way:

6/1982 – 3/1983, Position A;

4/1983 – 12/1983, Position B;

1/1984 – 8/1987, Position C;

...it would be better to show simply:

1982 – 1983, Position A;

1984 – 1987 Position C.

## IMPROVE YOUR COMMUNICATION SKILLS

Here are 6 great tips you can use:

1. Awareness of your own interaction with other people is the first step in improving your communication skills.

Learn to identify which types of situations make you uncomfortable and then modify your behavior to achieve positive results is a critical step in improving your communication skills.

You can learn to become aware of behaviors in other people that prompt you to respond in negative ways and modify your own behavior to turn the situation into a positive experience.

2. You must accept responsibility for your own behavior and do not fear apologizing for errors in judgment or insensitive actions.

Asking others for honest feedback about the way you interact with others can be very helpful. Accept the negative feedback along with the positive and make changes accordingly.

3. Your non-verbal communication is equally as important as the things that you say. Positive body language is extremely important in your interactions with other people.

If your words and your actions do not match, you will have a difficult time succeeding in social situations.

4. In order to learn how to improve your communication skills, you must become a great listener. You must fight the urge to respond immediately and really listen to what the other person is trying to communicate.

Offering suggestions or criticism before you are certain of the other person's intent can only lead to frustration for both parties.

5. Improving your communication skills is a process and cannot be accomplished overnight. Trying to improve or change too many things at once will be counter-productive.

You will become discouraged and overwhelmed if you attempt to change your entire personality all at once. Choose one or two traits at a time and work on those over a period of time. Learn to take advantage of your personal strengths and make a positive impact on others.

6. **Maximize your positive personality traits** and use them in your interactions with others. Good communication and great listening skills are the most important tools you can use in improving your communication skills.

You can learn how to improve your communication skills by developing excellent listening skills, learning to resolve problems and conflicts, understanding body language, and accepting responsibility for your own negative behavior.

Determination and self-awareness will make your desire to improve your communication skills a reality.

### Core JAVA Faq

1. what is a transient variable?  
A transient variable is a variable that may not be serialized.

2. which containers use a border Layout as their default layout?  
The window, Frame and Dialog classes use a border layout as their default layout.

3.Why do threads block on I/O?  
Threads block on i/o (that is enters the waiting state) so that other threads may execute while the i/o Operation is performed.

4. How are Observer and Observable used?  
Objects that subclass the Observable class maintain a list of observers. When an Observable object is updated it invokes the update() method of each of its observers to notify the observers that it has changed state. The Observer interface is implemented by objects that observe Observable objects.

5. What is synchronization and why is it important? With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

6. Can a lock be acquired on a class? Yes, a lock can be acquired on a class. This lock is acquired on the class's Class object.

7. What's new with the stop(), suspend() and resume() methods in JDK 1.2?

The stop(), suspend() and resume() methods have been deprecated in JDK 1.2.

8. Is null a keyword?

The null value is not a keyword.

9. What is the preferred size of a component?

The preferred size of a component is the minimum component size that will allow the component to display normally.

10. What method is used to specify a container's layout?

The setLayout() method is used to specify a container's layout.

11. Which containers use a FlowLayout as their default layout?

The Panel and Applet classes use the FlowLayout as their default layout.

12. What state does a thread enter when it terminates its processing?

When a thread terminates its processing, it enters the dead state.

13. What is the Collections API?

The Collections API is a set of classes and interfaces that support operations on collections of objects.

14. Which characters may be used as the second character of an identifier, but not as the first character of an identifier?  
The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.

15. What is the List interface?

The List interface provides support for ordered collections of objects.

16. How does Java handle integer overflows and underflows?  
It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

17. What is the Vector class?

The Vector class provides the capability to implement a growable array of objects

18. What modifiers may be used with an inner class that is a member of an outer class?

A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.

19. What is an Iterator interface?

The Iterator interface is used to step through the elements of a Collection.

20. What is the difference between the >> and >>> operators?

The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out.

21. Which method of the Component class is used to set the position and size of a component?

setBounds()

22. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8

characters?

Unicode requires 16 bits and ASCII require 7 bits. Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits. UTF-8 represents characters using 8, 16, and 18 bit patterns. UTF-16 uses 16-bit and larger bit patterns.

23. What is the difference between yielding and sleeping?

When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state.

24. Which java.util classes and interfaces support event handling?

The EventObject class and the EventListener interface support event processing.

25. Is sizeof a keyword?

The sizeof operator is not a keyword.

26. What are wrapped classes?

Wrapped classes are classes that allow primitive types to be accessed as objects.

27. Does garbage collection guarantee that a program will not run out of memory?

Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory

resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to

garbage collection

28. What restrictions are placed on the location of a package statement within a source code file?

A package statement must appear as the first line in a source code file (excluding blank lines and comments).

29. Can an object's finalize() method be invoked while it is reachable?

An object's finalize() method cannot be invoked by the garbage collector while the object is still reachable.

However, an object's finalize() method may be invoked by other objects.

30. What is the immediate superclass of the Applet class?

Panel

31. What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher

priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the

pool of ready tasks. The scheduler then determines which task should execute

next, based on priority and other factors.

32. Name three Component subclasses that support painting.

The Canvas, Frame, Panel, and Applet classes support painting.

33. What value does readLine() return when it has reached the end of a file?

The readLine() method returns null when it has reached the end of a file.

34. What is the immediate superclass of the Dialog class?

Window

35. What is clipping?

Clipping is the process of confining paint operations to a limited area or shape.

36. What is a native method?

A native method is a method that is implemented in a language other than Java.

37. Can a for statement loop indefinitely?

Yes, a for statement can loop indefinitely. For example, consider the following:

```
for(;;) ;
```

38. What are order of precedence and associativity, and how are they used?

Order of precedence determines the order in which operators are evaluated in expressions. Associativity determines whether an expression is evaluated left-to-right or right-to-left

39. When a thread blocks on I/O, what state does it enter?

A thread enters the waiting state when it blocks on I/O.

40. To what value is a variable of the String type automatically initialized?

The default value of an String type is null.

41. What is the catch or declare rule for method declarations?

If a checked exception may be thrown within the body of a method, the method must

either catch the exception or declare it in its throws clause.

42. What is the difference between a MenuItem and a CheckboxMenuItem?

The CheckboxMenuItem class extends the MenuItem class to support a menu item that may be checked or unchecked.

43. What is a task's priority and how is it used in scheduling?

A task's priority is an integer value that identifies the relative order in which it should be executed with respect to other

tasks. The scheduler attempts to schedule higher priority tasks before lower priority tasks.

44. What class is the top of the AWT event hierarchy?

The java.awt.AWTEvent class is the highest-level class in the AWT event-class hierarchy.

45. When a thread is created and started, what is its initial state?

A thread is in the ready state after it has been created and started.

46. Can an anonymous class be declared as implementing an interface and extending a class?

An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.

47. What is the range of the short type?

The range of the short type is -(2<sup>15</sup>) to 2<sup>15</sup> - 1.

48. What is the range of the char type?

The range of the char type is 0 to 2<sup>16</sup> - 1.

49. In which package are most of the AWT events that support the event-delegation model defined?

Most of the AWT-related events of the event-delegation model are defined in the java.awt.event package.

The AWTEvent class is defined in the java.awt package.

50. What is the immediate superclass of Menu?

MenuItem

51. What is the purpose of finalization?

The purpose of finalization is to give an unreachable object the opportunity to perform any cleanup processing before the

object is garbage collected.

52. Which class is the immediate superclass of the MenuComponent class.

Object

53. What invokes a thread's run() method?

After a thread is started, via its start() method or that of the Thread class, the JVM invokes the thread's run() method when the thread is initially executed.

54. What is the difference between the Boolean & operator and the && operator?

If an expression involving the Boolean & operator is evaluated, both operands are

evaluated. Then the & operator is applied to the operand. When an expression involving the && operator is evaluated, the

first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The

&& operator is then applied to the first and second operands. If the first operand

evaluates to false, the evaluation of the second operand is skipped.

55. Name three subclasses of the Component class.

Box.Filler, Button, Canvas, Checkbox, Choice, Container, Label, List, Scrollbar, or TextComponent

56. What is the GregorianCalendar class?

The GregorianCalendar provides support for traditional Western calendars.

57. Which Container method is used to cause a container to be laid out and redisplayed?

validate()

58. What is the purpose of the Runtime class?

The purpose of the Runtime class is to provide access to the Java runtime system.

59. How many times may an object's finalize() method be invoked by the garbage collector?

An object's finalize() method may only be invoked once by the garbage collector.

60. What is the purpose of the finally clause of a try-catch-finally statement?

The finally clause is used to provide the capability to execute code no matter whether or not an exception is thrown or caught.

61. What is the argument type of a program's main() method?

A program's main() method takes an argument of the String[] type.

62. Which Java operator is right associative?

The = operator is right associative.

63. What is the Locale class?

The Locale class is used to tailor program output to the conventions of a particular geographic, political, or cultural region.

64. Can a double value be cast to a byte?

Yes, a double value can be cast to a byte.

65. What is the difference between a break statement and a continue statement?

A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue

statement is used to end the current loop iteration and return control to the loop statement.

66. What must a class do to implement an interface?

It must provide all of the methods in the interface and identify the interface in its implements clause.

67. What method is invoked to cause an object to begin executing as a separate thread?

The start() method of the Thread class is invoked to cause an object to begin executing as a separate thread.

68. Name two subclasses of the TextComponent class.

TextField and TextArea

69. What is the advantage of the event-delegation model over the earlier event-inheritance model?

The event-delegation model has two advantages over the event-inheritance model. First, it enables event handling to be

handled by objects other than the ones that generate the events (or their containers). This allows a clean separation between

a component's design and its use. The other advantage of the event-delegation

model is that it performs much better in applications where many events are

generated. This performance improvement is due to the fact that the event-delegation model does not have to repeatedly

process unhandled events, as is the case of the event-inheritance model.

70. Which containers may have aMenuBar?

Frame

71. How are commas used in the initialization and iteration parts of a for statement?

Commas are used to separate multiple statements within the initialization and iteration parts of a for statement.

72. What is the purpose of the wait(), notify(), and notifyAll() methods?

The wait(), notify(), and notifyAll() methods are used to provide an efficient way for

threads to wait for a shared resource. When a thread executes an object's `wait()` method, it enters the waiting state. It only

enters the ready state after another thread invokes the object's `notify()` or `notifyAll()` methods.

73. What is an abstract method?

An abstract method is a method whose implementation is deferred to a subclass.

74. How are Java source code files named?

A Java source code file takes the name of a public class or interface that is defined within the file. A source code file may

contain at most one public class or interface. If a public class or interface is defined within a source code file, then the

source code file must take the name of the public class or interface. If no public class or interface is defined within a

source code file, then the file must take on a name that is different than its classes and interfaces. Source code files use

the `.java` extension.

75. What is the relationship between the `Canvas` class and the `Graphics` class?

A `Canvas` object provides access to a `Graphics` object via its `paint()` method.

76. What are the high-level thread states?

The high-level thread states are ready, running, waiting, and dead.

77. What value does `read()` return when it has reached the end of a file?

The `read()` method returns `-1` when it has reached the end of a file.

78. Can a `Byte` object be cast to a `double` value?

No, an object cannot be cast to a primitive value.

79. What is the difference between a static and a non-static inner class?

A non-static inner class may have object instances that are associated with instances of the class's outer class. A static

inner class does not have any object instances.

80. What is the difference between the `String` and `StringBuffer` classes?

String objects are constants. StringBuffer objects are not.

81. If a variable is declared as private, where may the variable be accessed?

A private variable may only be accessed within the class in which it is declared.

82. What is an object's lock and which object's have locks?

An object's lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may

execute a synchronized method of an object only after it has acquired the object's lock. All objects and classes have locks.

A class's lock is acquired on the class's Class object.

83. What is the Dictionary class?

The Dictionary class provides the capability to store key-value pairs.

84. How are the elements of a BorderLayout organized?

The elements of a BorderLayout are organized at the borders (North, South, East, and West) and the center of a container.

85. What is the % operator?

It is referred to as the modulo or remainder operator. It returns the remainder of dividing the first operand by the second

operand.

86. When can an object reference be cast to an interface reference?

An object reference be cast to an interface reference when the object implements the referenced interface.

87. What is the difference between a Window and a Frame?

The Frame class extends Window to define a main application window that can have a menu bar.

88. Which class is extended by all other classes?

The Object class is extended by all other classes.

89. Can an object be garbage collected while it is still reachable?

A reachable object cannot be garbage collected. Only unreachable objects may be garbage collected..

90. Is the ternary operator written  $x : y ? z$  or  $x ? y : z$  ?

It is written  $x ? y : z$ .

91. What is the difference between the Font and FontMetrics classes?

The FontMetrics class is used to define implementation-specific

properties, such as

ascent and descent, of a Font object.

92. How is rounding performed under integer division?

The fractional part of the result is truncated. This is known as rounding toward zero.

93. What happens when a thread cannot acquire a lock on an object?

If a thread attempts to execute a synchronized method or synchronized statement and is unable to acquire an object's lock, it

enters the waiting state until the lock becomes available.

94. What is the difference between the Reader/Writer class hierarchy and the

InputStream/OutputStream class hierarchy?

The Reader/Writer class hierarchy is character-oriented, and the

InputStream/OutputStream class hierarchy is byte-oriented.

95. What classes of exceptions may be caught by a catch clause?

A catch clause can catch any exception that may be assigned to the Throwable type. This includes the Error and Exception

types.

96. If a class is declared without any access modifiers, where may the class be accessed?

A class that is declared without any access modifiers is said to have package access. This means that the class can only be

accessed by other classes and interfaces that are defined within the same package.

97. What is the SimpleTimeZone class?

The SimpleTimeZone class provides support for a Gregorian calendar.

98. What is the Map interface?

The Map interface replaces the JDK 1.1 Dictionary class and is used to associate keys with values.

99. Does a class inherit the constructors of its superclass?

A class does not inherit constructors from any of its superclasses.

100. For which statements does it make sense to use a label?

The only statements for which it makes sense to use a label are those statements that can enclose a break or continue

statement.

101. What is the purpose of the System class?

The purpose of the System class is to provide access to system resources.

102. Which TextComponent method is used to set a TextComponent to the readonly state?

`setEditable()`

103. How are the elements of a CardLayout organized?

The elements of a CardLayout are stacked, one on top of the other, like a deck of cards.

104. Is `&&=` a valid Java operator?

No, it is not.

105. Name the eight primitive Java types.

The eight primitive types are byte, char, short, int, long, float, double, and boolean.

106. Which class should you use to obtain design information about an object?

The Class class is used to obtain information about an object's design.

107. What is the relationship between clipping and repainting?

When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires

repainting.

108. Is "abc" a primitive value?

The String literal "abc" is not a primitive value. It is a String object.

109. What is the relationship between an event-listener interface and an eventadapter class?

An event-listener interface defines the methods that must be implemented by an event handler for a particular kind of event.

An event adapter provides a default implementation of an eventlistener interface.

110. What restrictions are placed on the values of each case of a switch statement?

During compilation, the values of each case of a switch statement must evaluate to a value that can be promoted to an int value.

111. What modifiers may be used with an interface declaration?

An interface may be declared as public or abstract.

112. Is a class a subclass of itself?

A class is a subclass of itself.

113. What is the highest-level event class of the event-delegation model?

The java.util.EventObject class is the highest-level class in the event-delegation class hierarchy.

114. What event results from the clicking of a button?

The ActionEvent event is generated as the result of the clicking of a button.

115. How can a GUI component handle its own events?

A component can handle its own events by implementing the required event-listener interface and adding itself as its own event listener.

116. What is the difference between a while statement and a do statement?

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement

checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute

the body of a loop at least once.

117. How are the elements of a GridLayout organized?

The elements of a GridLayout are organized according to a grid.

However, the

elements are of different sizes and may occupy more than one row or column of the grid. In addition, the rows and columns may

have different sizes.

118. What advantage do Java's layout managers provide over

traditional windowing

systems?

Java uses layout managers to lay out components in a consistent manner across all

windowing platforms. Since Java's layout managers aren't tied to absolute sizing and positioning, they are able to accommodate

platform-specific differences among windowing systems.

119. What is the Collection interface?

The Collection interface provides support for the implementation of a mathematical bag -an unordered collection of objects

that may contain duplicates.

120. What modifiers can be used with a local inner class?

A local inner class may be final or abstract.

121. What is the difference between static and non-static variables?

A static variable is associated with the class as a whole rather than with specific instances of a class. Nonstatic variables

take on unique values with each object instance.

122. What is the difference between the paint() and repaint() methods?

The paint() method supports painting via a Graphics object. The repaint() method is used to cause paint() to be invoked by

the AWT painting thread.

123. What is the purpose of the File class?

The File class is used to create objects that provide access to the files and directories of a local file system.

124. Can an exception be rethrown?

Yes, an exception can be rethrown.

125. Which Math method is used to calculate the absolute value of a number?

The abs() method is used to calculate absolute values.

126. How does multithreading take place on a computer with a single CPU?

The operating system's task scheduler allocates execution time to multiple tasks. By quickly switching between executing tasks, it creates the impression that tasks execute sequentially.

127. When does the compiler supply a default constructor for a class?

The compiler supplies a default constructor for a class if no other constructors are provided.

128. When is the finally clause of a try-catch-finally statement executed?

The finally clause of the try-catch-finally statement is always executed unless the thread of execution terminates or an exception occurs within the execution of the finally clause.

129. Which class is the immediate superclass of the Container class?  
Component

130. If a method is declared as protected, where may the method be accessed?

A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

131. How can the Checkbox class be used to create a radio button?  
By associating Checkbox objects with a CheckboxGroup.

132. Which non-Unicode letter characters may be used as the first character of an identifier?

The non-Unicode letter characters \$ and \_ may appear as the first character of an identifier

133. What restrictions are placed on method overloading?  
Two methods may not have the same name and argument list but different return types.

134. What happens when you invoke a thread's interrupt method while it is sleeping or waiting?

When a task's interrupt() method is executed, the task enters the ready state. The next time the task enters the running

state, an InterruptedException is thrown.

135. What is casting?

There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller

values, such as byte values. Casting between object references is used to refer to an object by a compatible class,

interface, or array type reference.

136. What is the return type of a program's main() method?

A program's main() method has a void return type.

137. Name four Container classes.

Window, Frame, Dialog, FileDialog, Panel, Applet, or ScrollPane

138. What is the difference between a Choice and a List?

A Choice is displayed in a compact form that requires you to pull it down to see the list of available choices. Only one item

may be selected from a Choice. A List may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

139. What class of exceptions are generated by the Java run-time system?

The Java runtime system generates RuntimeException and Error exceptions.

140. What class allows you to read objects directly from a stream?

The ObjectInputStream class supports the reading of objects from input streams.

141. What is the difference between a field variable and a local variable?

A field variable is a variable that is declared as a member of a class. A local variable is a variable that is declared local

to a method.

142. Under what conditions is an object's finalize() method invoked by the garbage collector?

The garbage collector invokes an object's finalize() method when it detects that the object has become unreachable.

143. How are this() and super() used with constructors?

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

144. What is the relationship between a method's throws clause and the exceptions

that can be thrown during the method's execution?

A method's throws clause must declare any checked exceptions that are not caught within the body of the method.

145. What is the difference between the JDK 1.02 event model and the eventdelegation model introduced with JDK 1.1?

The JDK 1.02 event model uses an event inheritance or bubbling approach. In this model, components are required to handle

their own events. If they do not handle a particular event, the event is inherited by (or bubbled up to) the component's

ontainer. The container then either handles the event or it is bubbled up to

its container and so on, until the highest-level container has been tried.

In the event-delegation model, specific objects are designated as event handlers for GUI components. These objects implement

event-listener interfaces. The event-delegation model is more efficient than the event-inheritance model because it

eliminates the processing required to support the bubbling of unhandled events.

146. How is it possible for two String objects with identical values not to be equal

under the == operator?

The == operator compares two objects to determine if they are the same object in

memory. It is possible for two String objects to have the same value, but located in different areas of memory.

147. Why are the methods of the Math class static?

So they can be invoked as if they are a mathematical code library.

148. What Checkbox method allows you to tell if a Checkbox is checked?

getState()

149. What state is a thread in when it is executing?

An executing thread is in the running state.

150. What are the legal operands of the instanceof operator?

The left operand is an object reference or null value and the right

operand is a class,  
interface, or array type.

151. How are the elements of a GridLayout organized?

The elements of a GridLayout are of equal size and are laid out using the squares of a grid.

152. What an I/O filter?

An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

153. If an object is garbage collected, can it become reachable again?

Once an object is garbage collected, it ceases to exist. It can no longer become reachable again.

154. What is the Set interface?

The Set interface provides methods for accessing the elements of a finite mathematical set. Sets do not allow duplicate elements.

155. What classes of exceptions may be thrown by a throw statement?

A throw statement may throw any expression that may be assigned to the Throwable type.

156. What are E and PI?

E is the base of the natural logarithm and PI is mathematical value pi.

157. Are true and false keywords?

The values true and false are not keywords.

158. What is a void return type?

A void return type indicates that a method does not return a value.

159. What is the purpose of the enableEvents() method?

The enableEvents() method is used to enable an event for a particular object. Normally, an event is enabled when a listener

is added to an object for a particular event. The enableEvents() method is used by objects that handle events by overriding their event-dispatch methods.

160. What is the difference between the File and RandomAccessFile classes?

The File class encapsulates the files and directories of the local file system. The

RandomAccessFile class provides the methods needed to directly access data contained in any part of a file.

161. What happens when you add a double value to a String?  
The result is a String object.

162. What is your platform's default character encoding?

If you are running Java on English Windows platforms, it is probably Cp1252. If you are running Java on English Solaris

platforms, it is most likely 8859\_1..

163. Which package is always imported by default?

The java.lang package is always imported by default.

164. What interface must an object implement before it can be written to a stream  
as an object?

An object must implement the Serializable or Externalizable interface before it can be written to a stream as an object.

165. How are this and super used?

this is used to refer to the current object instance. super is used to refer to the variables and methods of the superclass

of the current object instance.

166. What is the purpose of garbage collection?

The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their

resources may be reclaimed and reused.

167. What is a compilation unit?

A compilation unit is a Java source code file.

168. What interface is extended by AWT event listeners?

All AWT event listeners extend the `java.util.EventListener` interface.

169. What restrictions are placed on method overriding?

Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides. The overriding method may not throw any exceptions

that may not be thrown by the overridden method.

170. How can a dead thread be restarted?

A dead thread cannot be restarted.

171. What happens if an exception is not caught?

An uncaught exception results in the `uncaughtException()` method of the thread's `ThreadGroup` being invoked, which eventually results in the termination of the program in which it is thrown.

172. What is a layout manager?

A layout manager is an object that is used to organize components in a container.

173. Which arithmetic operations can result in the throwing of an `ArithmaticException`?

`Integer /` and `%` can result in the throwing of an `ArithmaticException`.

174. What are three ways in which a thread can enter the waiting state?

A thread can enter the waiting state by invoking its `sleep()` method, by blocking on I/O, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's `wait()` method. It can also enter the waiting state by invoking its (deprecated) `suspend()` method.

175. Can an abstract class be final?

An abstract class may not be declared as final.

176. What is the ResourceBundle class?

The ResourceBundle class is used to store locale-specific resources that can be loaded by a program to tailor the program's appearance to the particular locale in which it is being run.

177. What happens if a try-catch-finally statement does not have a catch clause to handle an exception that is thrown within the body of the try statement?

The exception propagates up to the next higher level try-catch statement (if any) or results in the program's termination.

178. What is numeric promotion?

Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, byte, char, and short values are converted to int values. The int values

are also converted to long values, if necessary. The long and float values are converted to double values, as required.

179. What is the difference between a Scrollbar and a ScrollPane?

A Scrollbar is a Component, but not a Container. A ScrollPane is a Container. A ScrollPane handles its own events and performs its own scrolling.

180. What is the difference between a public and a non-public class?

A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.

181. To what value is a variable of the boolean type automatically initialized?

The default value of the boolean type is false.

182. Can try statements be nested?

Try statements may be tested.

183. What is the difference between the prefix and postfix forms of the `++` operator?

The prefix form performs the increment operation and returns the value of the increment operation. The postfix form returns

the current value all of the expression and then performs the increment operation on that value.

184. What is the purpose of a statement block?

A statement block is used to organize a sequence of statements as a single statement group.

185. What is a Java package and how is it used?

A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups

of classes and interfaces. Packages are also used to organize related classes and interfaces into a single API unit and to

control accessibility to these classes and interfaces.

186. What modifiers may be used with a top-level class?

A top-level class may be public, abstract, or final.

187. What are the `Object` and `Class` classes used for?

The `Object` class is the highest-level class in the Java class hierarchy. The `Class` class is used to represent the classes and

interfaces that are loaded by a Java program.

188. How does a try statement determine which catch clause should be used to

handle an exception?

When an exception is thrown within the body of a try statement, the catch clauses of the try statement are examined in the

order in which they appear. The first catch clause that is capable of handling the exception is executed. The remaining catch

clauses are ignored.

189. Can an unreachable object become reachable again?

An unreachable object may become reachable again. This can happen when the object's finalize() method is invoked and the

object performs an operation which causes it to become accessible to reachable objects.

190. When is an object subject to garbage collection?

An object is subject to garbage collection when it becomes unreachable to the program in which it is used.

191. What method must be implemented by all threads?

All tasks must implement the run() method, whether they are a subclass of Thread or implement the Runnable interface.

192. What methods are used to get and set the text label displayed by a Button object?

getLabel() and setLabel()

193. Which Component subclass is used for drawing and painting?

Canvas

194. What are synchronized methods and synchronized statements?

Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

195. What are the two basic ways in which classes that can be run as threads may be defined?

A thread class may be declared as a subclass of Thread, or it may implement the  
Runnable interface.

196. What are the problems faced by Java programmers who don't use layout

managers?

Without layout managers, Java programmers are faced with determining how their GUI will be displayed across multiple windowing systems and finding a common sizing and positioning that will work within the constraints imposed by each windowing system.

197. What is the difference between an if statement and a switch statement?

The if statement is used to select among two alternatives. It uses a boolean expression to decide which alternative should be executed. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.

198. What is the List interface?

The List interface provides support for ordered collections of objects.

#### Java Tips

- No explicit cast needed for upcasting, but explicit cast needed for downcasting.
- Defining your class as *implementing* an interface marks objects of that class as an *instance* of that interface.
- An *abstract* method cannot (obviously) be *final*.
- An *abstract* method cannot be *static* because *static* methods cannot be overridden.
- An instance method can be both *protected* and *abstract*. A static method can be *protected*.
- Before Java runtime clones an object, it checks to see if the object's class implements the Cloneable interface. If it does, the *clone()* method returns a clone of the object. If not, the *clone()* method throws a *CloneNotSupportedException*.

- The clone method is protected, so an object can only request a clone of another object which is either in the same package or which it inherits from. (i.e. standard meaning of protected)
- The JVM does not call an object's constructor when you clone the object.

## Keywords

- Classes can be modified from their default state using any of the three keywords: *public*, *abstract*, and *final*. So, can't have a static class, only static methods.
- A final variable is a constant, and a final method cannot be overridden.
- A synchronised method can belong to an object or to a class.
- Only one public class per file.
- package statement must come first in file and must be followed by any import statements.
- An identifier is an unlimited-length sequence of *Java letters* and *Java digits*, the first of which must be a Java letter. Java letters include \_ and \$. Digits include 0..9.

## Constructors

- The JVM does not call an object's constructor when an object is cloned.
- Constructors never return a value. If you do specify a return value, the JVM will interpret your intended constructor as a method.
- If a class contains no constructor declarations, then a *default* constructor that takes no arguments is supplied. This default constructor invokes the no-args superclass constructor, i.e. calls super();

- If, in a constructor, you do not make a direct call to *super* or *this* (with or without args) [note: must be on the first line] then *super*(no args) will first be invoked then any code in the constructor will be executed. See constructors.jpr project.
- A call to *this* in a constructor must also be on the first line. Note: can't have an explicit call to *super* followed by a call to *this* in a constructor - only one direct call to another constructor is allowed.

## Memory and Garbage Collection

- As soon as you lose the reference to an object, that object becomes eligible for garbage collection.
- Setting your object reference to *null* makes it a candidate for garbage collection.
- You can directly invoke the garbage collector by getting an object which represents the current runtime and invoking that object's *gc()* method (see p.95 of Exam Guide).
- Can't predict when garbage collection will occur, but it does run whenever memory gets low.
- If you want to perform some task when your object is about to be garbage collected, you can override the *java.lang.Object* method called *finalize()*. This method is declared as protected, does not return a value, and throws a *Throwable* object, i.e. *protected void finalize() throws Throwable*.
- Always invoke the superclass's *finalize()* method if you override *finalize()*.
- The JVM only invokes *finalize()* once per object. Since this is the case, do not resurrect an object in *finalize* as when the object is finalized again its *finalize()* method will not be called. Instead you should create a clone of the object if you must bring the object back to life.
- Remember that Java passes method parameters by value and not by reference. Obviously then, anything that happens to a primitive

data type inside a method does not affect the original value in the calling code. Also, any reassignment of object references inside a method has no effect on the objects passed in.

## Data Types and Values

- Ranges for primitive data types are as follows:-

| <i>Data Type</i> | <i>Range of Values</i> |                       |
|------------------|------------------------|-----------------------|
| byte             | -27.. 27-1             | signed integer        |
| short            | -215.. 215-1           | signed integer        |
| int              | -231.. 231-1           | signed integer        |
| long             | -263.. 263-1           | signed integer        |
| float            | 32 bit                 | signed floating point |
| double           | 64 bit                 | signed floating point |
| char             | 16 bit                 | Unicode character     |
| boolean          | either true or false   |                       |

- To specify an octal (base number, put a leading '0' in front of it)
- To specify a hexadecimal (base 16) number, put a leading '0x' in front of it.
- By default, integer values are of type int. However, you can force an integer value to be a long by placing an 'L' after it.
- By default, floating point values are of type double. However, you can force a floating-point value to be a float by placing an 'F' after it.

- Java sets each element in an array to its default value when it is created. Default object value is *null*, default integer value is 0, default boolean value is *false*, default floating point value is 0.0, default char value is '\u0000'.
- Arrays are objects allocated at runtime, so you can use a variable to set their length.
- First element in an array is at index 0 and last element is at *length*-1. *length* is a special array variable (not a method, so don't need round brackets after it).
- Can only use curly braces in array initialisation when array is actually declared, i.e. can't declare the array on one line then initialise it with curly braces on line below.
- ASCII characters are all found in the range '\u0000' to '\u00ff'
- The default value for any class variable or instance variable declared as a char is '\u0000'.

## Operators

- >> is right shift keep the sign, >>> is right shift don't keep the sign.
- & and | can be used with both integral and boolean types. If used with integers the result is integral. If used with booleans, both operands are evaluated, even when the result of the operation can be determined after evaluating only the left operand.
- && and || are used with boolean operands only. The right operand is not evaluated if the result of the operation can be determined after evaluating only the left operand.
- The *equals()* method (defined at the 'highest' level as a method of Object) is used to test the value of an object. The == operator is used to test the object references *themselves*.
- By default, *equals()* returns true only if the objects reside in the same memory location, i.e. if the object references are equal. So, by

default, equals() and == do the same things. *This will be the case for all classes that do not override equals()*.

- String, Wrappers (including Integer, Long, Float, Double, Character and Boolean), BitSet, Date and File classes all override equals() so that the value true is returned if the values are equal.
- The + and += operators are overloaded for Strings.
- Dividing an integer by 0 is illegal and would cause Java to throw an *ArithmException*. Floating point numbers have values for infinity and not-a-number, so using arithmetic operators on floating point numbers never results in an exception.

## Control Flow

- A loop counter is usually an integer, however it could also be a floating point number - incrementing by 1.0
- Breaking to a label means that the loop at the label is terminated. Any outer loop will keep iterating.
- In contrast, a continue to a label continues execution with the next iteration of the labelled loop.
- The expression for an if and while statement *must be a Boolean*.
- The expression in a switch statement *must be an int or a char*.
- A default statement in a switch is optional.
- A case block will fall through to the case block which follows it, unless the last statement in a case block is a *throw, return or break*.
- As with if statements, for and while loops, it is possible to nest switch statements.
- Any line of code can be labelled BUT can only do a labelled continue to a loop, and can only do a labelled break to a loop or to an enclosing statement.

## Exceptions

- It is possible to have multiple catch blocks in a try-catch-finally. The finally block is optional.
- A catch block must always be associated with a try block, i.e. can't have a catch block by itself or with a finally block.
- A finally block must always be associated with a try block, i.e. can't have a finally block by itself or with a catch block.
- With multiple catch blocks, the type of exception caught must progress from the most specific exception that you wish to catch to the superclasses for these exceptions. (Makes sense!)
- Methods *must* declare any exception which they throw.
- Invoking a method which declares it throws exceptions is not possible unless either the code is placed in a try-catch, or the calling method declares that it throws the exceptions, i.e. checked exceptions must be caught or rethrown. If the try-catch approach is used, then the try-catch must cope with all of the exceptions which a method declares it throws.
- You can list more than one exception in the throws clause if you separate them with commas.
- RuntimeException and its subclasses are *unchecked* exceptions.
- Unchecked exceptions do not have to be caught.
- All Errors are unchecked.
- You should never throw an unchecked exception in your own code, even though the code will compile.
- You cannot use a try block on its own. It must be accompanied by a following catch or finally (or both).
- Code in a finally block will always be executed, whether an exception is thrown or not and whether any exception thrown is caught or not. Only terminating the program will stop the finally code from being executed.

- ❑ A method can only throw those exceptions listed in its throws clause, or subclasses of those exceptions.
- ❑ A method can throw any unchecked exception, even if it is not declared in its throws clause.
- ❑ When you override a method, you must list those exceptions that the override code might throw. You can list only those exceptions, or subclasses of those exceptions, that are defined in the method definition you are inheriting from, i.e. you cannot add new exception types to those you inherit. You can choose to throw a subset of those exceptions listed in the method's superclass, however, a subclass further down the hierarchy cannot then re-list the exceptions dropped above it.

## Methods

- ❑ If you define more than one method with the same name in the same class, Java must be able to determine which method to invoke based on the number and types of parameters defined for that method.
- ❑ The compiler will complain if you have two methods with identical signatures exception for the return type and/or the exceptions thrown, i.e. can't overload based on return type and/or exceptions thrown.
- ❑ Determining which methods will be invoked with integer params - see p.195 of the Exam Guide.
- ❑ It is possible to declare an inherited method *abstract*.
- ❑ Obviously, it is not possible to override a *final* method.
- ❑ A subclass may make an inherited method *synchronized*, or it may leave off the synchronized keyword so that its version is not synchronized. If a method in a subclass is not synchronized but the method in the superclass is, the thread obtains the monitor for the object when it enters the superclass's method.

- ❑ It is possible to declare an inherited method as *abstract*, but then there is no way to get to the behaviour in the hierarchy above the *abstract* declaration.
- ❑ Return types must match the overridden method in the superclass exactly. The parameter types must match those in the superclass exactly, i.e. in the same order. If this is not the case, then the superclass's method is not overridden. Compiler will not complain, however, unless exactly the same signature except for return type (or exceptions thrown - see earlier).
- ❑ You cannot make a method in a subclass more private than it is defined in the superclass, 'though you can make it more public.'
- ❑ Coercion of arguments only happens with overloading, not overriding.
- ❑ It is perfectly legal to have two different instance variables with the same name if they are defined in different classes where one class inherits from the other.
- ❑ Which variable is accessed depends on the type of object reference which the variable was declared to hold. Which method gets invoked depends on the underlying object. See p.106 Q1 & p.201 of Exam Guide.
- ❑ A *native* method does not have a body, or even a set of braces, e.g. `public native void method();`
- ❑ A *native* method cannot be abstract.
- ❑ A *native* method can throw exceptions.

## Math and String Classes

- ❑ `ceil()` returns the next highest integer (expressed as a double)



| <i>Method call</i> | <i>Returns</i> |
|--------------------|----------------|
| ceil(9.01)         | 9.0            |
| ceil(-0.1)         | 0.0            |
| ceil(100)          | 100.0          |

- *round()* returns the closest integer (expressed as an int if the parameter was a float, or a long if the parameter was a double)

| <i>Method call</i> | <i>Returns</i> |
|--------------------|----------------|
| round(9.01)        | 9              |
| round(9.5)         | 10             |
| round(-9.5)        | -9             |
| round(-0.1)        | 0              |
| round(100)         | 100            |

*random()* returns a random number, a double, between 0.0 and 1.0

*sqrt()* takes a double and returns a double. If the argument is NaN or <0, the result is NaN.

*sin(), cos() and tan()* all take a double and return a double.

All objects respond to *toString()*, which you can override to return a String representation of your object.

String objects have the following methods: *length()*, *toUpperCase()*, *toLowerCase()*, *equals()*, *equalsIgnoreCase()*, *charAt()*, *indexOf()*, *lastIndexOf()*, *substring()*, *toString()* and *trim()*.

*There are four versions of indexOf()*

| Arguments                              | Results  |
|--|--|
| (int ch) - works if you put a char in! | Finds the first occurrence of this character                         |
| (int ch, int fromIndex)                | Finds the first occurrence of this character starting from fromIndex |
| (String substring)                     | Finds the start of this substring                                    |
| (String substring, int fromIndex)      | Finds the start of this substring searching from fromIndex           |

- There are four versions *lastIndexOf()* - as above table but, in each method, finding the last index rather than the first index.
- First character in a String is at position 0, last character is at position *length()-1*
- There are two versions of *substring()*

| Arguments        | Results   |
|------------------|---|
| (int startIndex) | Returns a substring starting with startIndex and extending to the end of the String |

(int startIndex, int  
endIndex)

Returns a substring  
starting with startIndex  
and extending to (but  
not including)  
endIndex

❑ For a String *toString()* returns itself - the same object reference (obviously pointing to the same object) that was used to invoke the method.

❑ *trim()* returns a new String object that cuts off any leading and trailing whitespace for the String for which it was invoked.

❑ Java considers “whitespace” to be any character with a code less than or equal to ‘\u0020’ which is the space character.

- Input/Output**
- ❑ InputStream and OutputStream are abstract classes.
  - ❑ FileInputStream and FilterInputStream both inherit directly from InputStream.
  - ❑ FileOutputStream and FilterOutputStream both inherit directly from OutputStream.
  - ❑ DataInputStream inherits directly from FilterInputStream.
  - ❑ DataOutputStream inherits directly from FilterOutputStream.
  - ❑ When you create a Filter stream, you must specify the stream to which it will attach.
  - ❑ A Filter stream processes a stream of bytes in some way. By “chaining” any number of Filter streams, you can add any amount of processing to a stream of bytes.
  - ❑ DataInputStream, DataOutputStream and RandomAccessFile know how to work with Java data types because they implement the

DataInput and DataOutput interfaces, whereas FileInputStream and FileOutputStream know only how to work with individual bytes.

- RandomAccessFile implements both DataInput and DataOutput methods - RandomAccessFile objects can read from and write to files.
- The File class is not used to create files. Can create a file using an instance of class RandomAccessFile and FileOutputStream.
- To test if a File object refers to an existing file, you can invoke `exists()` which returns true or false. The File methods `canRead()` and `canWrite()` return boolean values that indicate whether the application can read from or write to the file. (Note: applets can't write to a file.)
- Can use File methods to make a permanent change to the file system. For example, you can call `delete()` or `rename()`. For `rename()`, need to supply a File object which embodies the new name.
- You can create a directory using the File class. The method `mkdir()` does this.
- It is possible to navigate the filing system using the File class. Methods `getParent()`, `getPath()` and `getName()` are provided, and also `getAbsolutePath()`.
- The method `getAbsolutePath()` returns the name of the current user directory (the full pathname included) with the file name concatenated. See Practice Exam 1 Q.45
- Creating a FileOutputStream object creates the appropriate file. If the file already exists, FileOutputStream replaces it (unless the FileOutputStream object is created using the constructor which takes a String and a boolean - see later)
- Upon creation of a RandomAccessFile object, need to supply a file object plus a mode. Alternatively, can supply a filename plus a mode.
- Valid modes for RandomAccessFile are "r" and "rw".

- Constructors for FileInputStream: one takes a String, one takes a File, one takes a FileDescriptor
- Constructors for FileOutputStream: one takes a String, one takes a File, one takes a FileDescriptor, one takes a String and a boolean (which indicates whether or not to append).
- Constructors for FilterInputStream: one only, which takes an InputStream
- Constructors for FilterOutputStream: one only, which takes an OutputStream
- See Chapter 11 Review Questions for io stuff - very useful.

## Threads

- A Java program runs until the only threads left running are daemon threads.
- A Thread can be set as a user or daemon thread when it is created.
- In a standalone program, your class runs until your *main()* method exists - unless your *main()* method creates more threads.
- You can initiate your own thread of execution by creating a Thread object, invoking its *start()* method, and providing the behaviour that tells the thread what to do. The thread will run until its *run()* method exists, after which it will come to a halt - thus ending its life cycle.
- The Thread class, by default, doesn't provide any behaviour for *run()*.
- There are two ways to provide the behaviour for a thread
  - Subclass the thread and override the *run()* method - see p.253 of Exam Guide.
  - Implement the Runnable interface and indicate an instance of this class will be the thread's target.

- A thread has a life cycle. Creating a new Thread instance puts the thread into the “new thread” state. When the `start()` method is invoked, the thread is then “alive” and “runnable”. A thread at this point will respond to the method `isAlive()` by returning true.
- The thread will continue to return true to `isAlive()` until it is “dead”, no matter whether it is “runnable” or “not runnable”.
- If 2 threads are alive, with the same highest priority, the JVM switches between them. The JVM will switch between any number of threads with the same highest priority.
- The priority numbers for threads falls between the range of `Thread.MIN_PRIORITY` and `Thread.MAX_PRIORITY`.
- The default thread priority is `Thread.NORM_PRIORITY`.
- New threads take on the priority of the thread that spawned them.
- You can explicitly set the priority of a thread using `setPriority()`, and you can get the priority of a thread using `getPriority()`. There is no constructor for thread which takes a priority.
- The JVM determines the priority of when a thread can run based on its priority ranking, but this doesn’t mean that a low priority thread will not run.
- The currently executing thread can yield control by invoking `yield()`. If you invoke `yield()`, Java will pick a new thread to run. However, it is possible the thread that just yielded might run again immediately if it is the highest priority thread.
- There are 3 types of code that can be *synchronized*: class methods, instance methods, any block of code within a method.
- Variables cannot take the *synchronized* keyword.
- Synchronization stays in effect if you enter a synchronized method and call out to a non-synchronized method. The thread only gives up the monitor after the synchronized method returns.

- ❑ Using a thread's `stop()` method will make it die.
- ❑ There are 3 ways to transition a thread between "runnable" and "not runnable"

put it to sleep and wake it up  
pause it and resume it  
use the methods `wait()`, `notify()` and `notifyAll()`  
- these are methods of class `Object`. `wait()` throws `InterruptedException`.

❑ The third method listed above allows communication between the threads, whereas the other 2 methods don't.

❑ By using the methods `wait()`, `notify()` and `notifyAll()`, any thread can wait for some condition in an object to change, and any thread can notify all threads waiting on that object's condition that the condition has changed and that they should continue.

❑ When a waiting thread pauses, it relinquishes the object's monitor and waits to be notified that it should try to reacquire it.

❑ If you know you only have one thread waiting on a condition, you can feel free to use `notify()`, otherwise you should use `notifyAll()`. The `notifyAll()` method wakes up all threads waiting to reacquire the monitor for the object.

❑ The reasons why a thread might be "alive" and "not runnable" are as follows:

- ❑ the thread is not the highest priority thread and so is not able to get CPU time
- ❑ the thread has been put to sleep using the `sleep()` method (of class `Thread`. Throws `InterruptedException`)
- ❑ the thread has been suspended using the `suspend()` method

- ❑ the thread is waiting on a condition because someone invoked `wait()` for the thread
- ❑ the thread has explicitly yielded control by invoking `yield()`

## Graphical User Interfaces

- ❑ If you want to explicitly repaint a component, you should not call `paint()` directly. Instead you should invoke your component's `repaint()` method.
- ❑ The `repaint()` method is overloaded. The no-args version of `repaint()` does not cause your user interface to repaint right away. In fact, when `repaint()` returns, your component has not yet been repainted - you've only issued a request for a `repaint()`. There is another version of `repaint()` that requests the component to be repainted within a certain number of milliseconds.
- ❑ The `repaint()` method will cause AWT to invoke a component's `update()` method. AWT passes a `Graphics` object to `update()` - the same one that it passes to `paint()`. So, `repaint()` calls `update()` which calls `paint()`.
- ❑ The `Graphics` object that AWT hands to `update()` and `paint()` is different every time the Component is repainted.
- ❑ When Java repaints on its own, such as when the user resizes an applet, the AWT does not invoke `update()` - it just calls `paint()` directly.
- ❑ The `update()` method does three things in this order:

- clears the background of the object by filling it with its background colour
- sets the current drawing colour to be its foreground colour
- invokes *paint()*, passing it the Graphics object received
- Common graphics methods:
  - *drawString()* - takes a String, followed by x then y co-ord of baseline for first char
  - *drawLine()* - takes x then y co-ords of starting point, then x then y co-ords of end point
  - *drawRect() / fillRect()* - take 4 params - x then y co-ords of top left corner, then width then height
  - *drawOval() / fillOval()* - take 4 params - x, y, width, height and draw an oval inside corresponding rectangle
  - *drawPolygon() / fillPolygon()* - (int[] xpoints, int[] ypoints, int npoints) - array of x co-ords then array of y co-ords followed by the number of points to use
  - *drawArc() / fillArc()* - takes 6 params - x then y co-ord of top left corner (of bounding rectangle of corresponding oval), width then height of arc, then start angle (not really an angle, more a position on the clock - 0 represents 3 o'clock), then number of degrees to move along the arc. Last parameter can also be negative, if you want to move in a clockwise direction round the arc.
- The Image class does not provide a constructor for specifying where an image will come from.
- You can obtain an Image, typically by downloading it from the Internet by specifying a URL. You can retrieve an image and obtain an Image object by invoking an Applet method named *getImage()*. This method takes a URL.

- When the `getImage()` method returns, the data for the image is not necessarily immediately available. The method returns right away, even if the image resource is located over the Internet on a Web server and must be downloaded.
- When you invoke the Graphics method `drawImage()` the image download begins. The object you specify as an ImageObserver keeps the Java graphics system up-to-date on the state of the download. When the ImageObserver sees that the download is complete, it notifies the graphics system that it can draw the image in its entirety.
- ImageObserver is an interface. The Component class implements this interface, so any component (such as an Applet itself) can be used as an ImageObserver.
- TextArea takes rows then cols in constructor. If you type in more text than can be displayed all at once, scrollbars appear automatically.
- TextField takes only cols. A user may type one line into a TextField. No scrollbars appear if the line is too long to be displayed, but can use arrow keys to move to beginning or end of text.
- TextArea and TextField both inherit from TextComponent which provides a `setEditable()` method.
- For a variable width font, TextArea / TextField width is based on average of letter widths.
- Can populate a List using `addItem()` method.
- As well as a no-args constructor and a constructor which takes number of rows, List class also has a constructor which takes number of rows followed by a boolean value which represents whether or not multiple list item selections are allowed.

## Java 1.0

- o Component methods for Java 1.0.2 - *enable()*, *disable()*, *show()*, *hide()*, *size()*, *resize()*, *setForeground()*, *setBackground()*.
- o *enable()* / *disable()* - make a component selectable or not by the user.
- o *size()* / *resize()* - the *size()* method retrieves the size of the Component as a Dimension object which has two fields (*width* and *height*). The *resize()* method sets the size of the Component. The method is overloaded - one version takes a Dimension object and the other takes width and height directly as int values.
- o *show()* / *hide()* - the *show()* method makes a Component visible, the *hide()* method makes a Component invisible. The *show()* method is overloaded so that you can supply a boolean parameter to indicate whether to show the Component (if the parameter is true).
- o A Frame is only visible when the *show()* method has been invoked.

## Java 1.1

- o Component methods for Java 1.1 - *setEnabled()*, *getSize()*, *setSize()*, *setVisible()*, *setForeground()*, *setBackground()*. *setEnabled()* replaces *enable()* / *disable()* and takes a boolean value. *getSize()* and *setSize()* replace *size()* / *resize()*. *setVisible()* replaces *show()* / *hide()* and takes a boolean value.

- Java provides a Color class which defines many static constants that contain instances of class Color already initialized to common colours. To use red, for example, you can access Color.red. The Color class has a constructor which takes three int values of red, green and blue so that you can make up your own colour.
- Each container has exactly one layout manager which determines how to arrange the Components within a Container.
- A layout manager is any class that implements the LayoutManager interface. Java's 5 layout managers all inherit directly from class

Object, but they implement the LayoutManager interface, which defines 5 *abstract* methods:

- *addLayoutComponent()*
- *layoutContainer()*
- *minimumLayoutSize()*
- *preferredLayoutSize()*
- *removeLayoutComponent()*

□ Instead of invoking the layout manager's methods yourself, Java's default Container methods invoke them for you at the appropriate times. These Container methods include *add()*, *getMinimumSize()*, *getPreferredSize()*, *remove()* and *removeAll()* (these are all Java 1.1 methods).

□ Each type of container comes with a default layout manager. Default for a Frame, Window or Dialog is *BorderLayout*. Default for a Panel is *FlowLayout()*.

□ A *FlowLayout* object arranges components left to right and top to bottom, centering each line as it goes. The *FlowLayout* layout manager is the *only* layout manager which allows components to be their preferred size. If a container using a *FlowLayout* is resized, all of the components inside it might need to be rearranged, and some might not be completely visible.

□ A *BorderLayout* object arranges components according to the directions "North", "South", "East" and "West". There's also an area for "Center" which includes any space left over from other regions.

□ Components are rarely allowed to be their preferred size in a *BorderLayout*. *BorderLayout* stretches components. Stretches "North" and "South" components horizontally. Stretches "East" and "West" components vertically. Stretches "Center" components both horizontally and vertically.

- If nothing is placed in “East” or “West”, for example, then the “Center” stretches all the way from the left edge of the Container to the right edge.
- Some components are stretchable and others are not. For example, Button, Label and TextField are stretchable, whereas Checkbox is not.
- GridLayout objects allow you to specify a rectangular grid in which to place the components. Each cell in the grid is the same height as the other cells, and each width is the same as the other cells. Components are stretched both horizontally and vertically to fill the cell.
- When a GridLayout object is first constructed, you must first specify how many rows and how many columns the grid will have. Components are added to the grid left to right and top to bottom. If more components are added than there are columns, the GridLayout keeps the same number of rows but adds the necessary number of columns.
- You can change a Container’s layout manager to be a different one by invoking *setLayout()*.

## Event Handling

- All of the events dispatched by AWT’s components use event classes that are subclasses of AWTEvent, which is in java.awt. These subclasses are defined in the java.awt.event package.
- The AWT does not notify your event handler of every event that occurs over a component, as in the old days of 1.0.2. Now, AWT informs your event handler only about the events it is interested in.
- Define a class which implements the necessary Listeners and then use the Component *addABCListener()* method to register an interest in a certain type of event, e.g. *addMouseListener(this)*.
- The Listener interfaces inherit directly from java.util.EventListener.
- The Listener interfaces are

- ❑ ActionListener
  - ❑ AdjustmentListener
  - ❑ ComponentListener
  - ❑ FocusListener
  - ❑ ItemListener
  - ❑ KeyListener
  - ❑ MouseListener
  - ❑ MouseMotionListener
  - ❑ TextListener
  - ❑ WindowListener
- ❑ If you define a class which implements a listener, you must obviously include stubs for those listener methods not implemented as interfaces are implicitly *abstract* and failure to implement all of the methods would result in an abstract subclass.
- ❑ The Listener methods are as follows: (Note - all are declared as *public void* and all take only one parameter which is an Event of the same name as the Listener, i.e. ActionListener methods take an ActionEvent etc.

| Listener           | Methods             |
|--------------------|---------------------|
| ActionListener     | actionPerformed     |
| AdjustmentListener | adjustmentPerformed |
| ComponentListener  | componentHidden     |

|                     |  |
|---------------------|--|
|                     | componentMoved<br>componentResized<br>componentShown                         |
| ContainerListener   | componentAdded<br>componentRemoved   |
| FocusListener       | focusGained<br>focusLost   |
| ItemListener        | itemStateChanged   |
| KeyListener         | keyPressed<br>keyReleased<br>keyTyped  |
| MouseListener       | mouseClicked<br>mouseEntered<br>mouseExited<br>mousePressed<br>mouseReleased |
| MouseMotionListener | mouseDragged<br>mouseMoved   |
| TextListener        | textValueChanged   |
| WindowListener      | windowClosed   |

|  |                   |
|--|-------------------|
|  | windowClosing     |
|  | windowDeiconified |
|  | windowActivated   |
|  | windowDeactivated |
|  | windowIconified   |
|  | windowOpened      |

- There's an Adapter class to match each Listener interface. Each Adapter class defines no-op stubs for the methods declared in the corresponding interface. So can extend an Adapter rather than implement a Listener interface (but this only makes sense when the object that will listen for and handle the event has no other responsibilities).

### Passing Arguments to Programs

- The *main()* method must be declared as a public, static method that does not return a value and takes an array of String objects. The order of public and static, and the way in which the String array is defined, is not strictly enforced.
- When *main()* ends, that may or may not be the end of the program. The JVM will run until the only remaining threads are daemon threads. If *main()* does not spawn any more threads, then the program will end when *main()* ends.

### Embedding Applets into Web Pages

- The applet tag requires three codewords - *code*, *width* and *height*, e.g. <applet code=Metric.class width=200 height=100></applet>
- If the applet is loaded relative to the page, there are two things that can change the base location - one of which is if the HTML file itself contains a <base> tag - this tag specifies where to look for the applet class.

- The conventional way to pass a parameter to an applet is to use the <param> tag, using one <param> tag per parameter. Each <param> tag can take only one parameter.
- Each parameter value can be retrieved by the applet as a String, using methods defined in class Applet.
- If you want to pass a different type of value, such as an int or a boolean, you must convert it from a String, most likely by using a wrapper class.
- If you'd like a parameter to contain spaces, you should place this value between quotes. Otherwise it's not strictly necessary to pass params in quotes (good style though).
- You retrieve the value of a parameter using the *getParameter()* method defined by the Applet class. This method returns a String containing the value of the parameter, or null if the parameter was not defined at all.
- Class Applet inherits from Panel.
- When an applet instance is first instantiated, Java invokes the applet's init() method.
- When the Web page containing the applet is about to appear, Java invokes the applet's start() method.
- When the Web page is about to be replaced with another page, Java invokes the applets stop() method.
- When the Web page is removed from the browser's cache and the applet instance is about to go away, Java invokes the applet's destroy() method.

## Inner Classes

- In Java 1.1 you can define classes inside classes - these are known as "inner classes".

- ❑ If you define an inner class at the same level as the enclosing class' instance variables, the inner class can access those instance variables - no matter what their access control.
- ❑ If you define an inner class within a method, the inner class can access the enclosing class' instance variables and also the local variables and parameter for that method.
- ❑ If you do reference local variables or parameters from an inner class, those variables or parameters must be declared as *final* to help guarantee data integrity.
- ❑ You can also define an anonymous class - a class without a name. See p.399 Exam Guide for syntax.
- ❑ If you'd like to refer to the current instance of the enclosing class, you can write EnclosingClassName.this.
- ❑ If you need to refer to the inner class using a fully qualified name, you can write EnclosingClassName.InnerClassName.
- ❑ If your inner class is not defined as *static* you can only create new instances of this class from a non-*static* method.
- ❑ Anonymous classes cannot have const

## JAR Files

- ❑ JAR stands for "Java Archive". The 1.1 JDK comes with a utility called 'jar' that creates JAR files. Example syntax: jar -cf Jungle.jar Panther.class Leopard.class
- ❑ Directories are processed recursively
- ❑ Can specify a JAR file in a Web page by adding 'archive="jars/Jungle.jar"' to the <applet> tag. E.g. <applet code=ExampleApplet.class archive="jars/example.jar" width=200 height=110> </applet>

- You can also define more than one JAR file in the archive value by listing them all within the quotes and separating their names with commas.

## Serializing Objects

- In Java 1.1 it is now possible to read and write objects as well as primitive data types, using classes that implement ObjectInput and ObjectOutputStream. These two interfaces extend DataInput and DataOutput to read or write an object. ObjectInputStream and ObjectOutputStream implement these interfaces.
- If a class implements the Serializable interface, then its *public* and *protected* instance variables will be read from and written to the stream automatically when you use ObjectInputStream and ObjectOutputStream.
- If an instance variable refers to another object, it will also be read or written, and this continues recursively.
- If the referenced object does not implement the Serializable interface, Java will throw a *NotSerializableException*.
- The Serializable interface serves only to identify those instances of a particular class that can be read from and written to a stream. It does not actually define any methods that you must implement.
- If you create your own class and want to keep certain data from being read or written, you can declare that instance variable using the *transient* keyword. Java skips any variable declared as *transient* when it reads and writes the object following the Serializable protocol.

## Reflection

- Using the Reflection API, a program can determine a class' accessible fields, methods and constructors at runtime.

## Jdk 1.6 new features

Java Platform, Standard Edition (Java SE 6) which code-named “Mustang” is the new name and future version for what previously known as J2SE. The new release will be identified as product version 6, and developer version 1.6.0. Java SE 6 has two products delivered under the name of the platform, that's Java SE Development Kit 6 (JDK 6) and Java SE Runtime Environment 6 (JRE 6). Java SE 6 is currently in Beta 2 and is scheduled to be officially released later this year. Java SE 6 has greater level of maturity, stability, scalability and security of Java implementation.

Java SE 6 has a lot of new features, enhancements and improvements especially better GUI performance and better handling of behaviour of GUI applications, plus improvements and new features in server-side core and Java core. Some of the features and enhancements listed below:

- New `java.awt.Desktop` API package that offers the ability to easily integrate an application with other applications.
- Improved internationalization.
- JFC and Swing integration with desktop by using Windows API.
- Java 2D integration with desktop such as using desktop anti-aliasing font settings.
- Splash screen direct support and can be shown before JVM started.
- System tray support with ability to add icons, tool tips, and pop-up menus to the Windows or any other system tray (such as Gnome).
- New Java Compiler API.
- New Java Scripting framework.
- Enhanced Java Platform Debugger Architecture.
- Improved JMX Monitoring API to allows sending events when MBean attribute values pass specified thresholds.
- Hotspot JVM for Solaris provides hooks for the Solaris DTrace system debugging utility.
- Enhanced memory leak analysis and detection.
- New JVM option allows running of script when the heap is full.
- Integration with GSS/Kerberos.
- New Security Request Framework enable requesting of certificates over a number of protocols.
- Improved web services.

- Improved database connectivity such as JDBC 4.0 and automatic java.sql.Driver discovery.
- Improvements to the JVM Tool interface.

**For mock exams**

<http://www.danchisholm.net/july21/topic/index.html>

**for assertion tutorial and programs**

<http://java.sun.com/j2se/1.4.2/docs/guide/lang/assert.html>

**for threads**

[http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?ubb=get\\_topic&f=27&t=002122](http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?ubb=get_topic&f=27&t=002122)

**for all types of questions**

<http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi>

**for discussions**

<http://www.examulator.com/moodle/mod/forum/discuss.php?d=52>

**for all mock exam sites**

<http://www.akgupta.com/Java/links.htm>

### Question 1

```
class MWC101 {  
    public static void main(String[] args) {  
        int[] a1 = new int[];          // 1  
        int a2[] = new int[5];         // 2  
        int[] a3 = new int[]{1,2};      // 3  
        int []a4 = {1,2};             // 4  
        int[] a5 = new int[5]{1,2,3,4,5}; // 5  
    } }
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

### Question 2

```
class MWC102 {  
    public static void main (String[] args) {  
        byte[] a = new byte[1]; long[] b = new long[1];  
        float[] c = new float[1]; Object[] d = new Object[1];  
        System.out.print(a[0]+","+b[0]+","+c[0]+"," +d[0]);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,null
- b. Prints: 0,0,0,0,null
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 3

```
class MWC103 {  
    public static void main(String[] args) {  
        int[] a1 = {1,2,3};  
        int []a2 = {4,5,6};  
        int a3[] = {7,8,9};  
        System.out.print(a1[1]+","+a2[1]+","+a3[1]);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 12
- b. Prints: 15
- c. Prints: 18
- d. Prints: 1,4,7
- e. Prints: 2,5,8
- f. Prints: 3,6,9
- g. Compile-time error
- h. Run-time error
- i. None of the above

#### Question 4

```
class MWC104 {  
    public static void main(String[] args) {  
        int[5] a1; // 1  
        int []a2; // 2  
        int[ ]a3; // 3  
        int a4[]; // 4  
    } }
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

#### Question 5

```
class MWC105 {  
    static boolean b1;  
    public static void main(String[] args) {  
        boolean[] array = new boolean[1];  
        boolean b2;  
        System.out.print(b1+"," );  
        System.out.print(array[0]+"," );  
        System.out.print(b2);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: true,true,true
- b. Prints: false,false,false
- c. Prints: null,null,null
- d. Prints: false,true,false

- e. Compile-time error
- f. Run-time error
- g. None of the above

### Answers

| No. | Answer                     | Remark   |
|-----|----------------------------|--|
| 1   | a<br>e 1 5                 | An array creation expression must have either a dimension expression or an initializer. If both are present, then a compile-time error is generated. Similarly, if neither is present, then a compile-time error is generated. If only the dimension expression is present, then an array with the specified dimension is created with all elements set to the default values. If only the initializer is present, then an array will be created that has the required dimensions to accommodate the values specified in the initializer. Java avoids the possibility of an incompatible dimension expression and initializer by not allowing both to appear in the same array creation expression. A compile-time error is generated by the array creation expression for a1, because it needs either a dimension expression or an initializer. A compile-time error is generated at 5, because either the dimension expression or the initializer must be removed. |
| 2   | b Prints:<br>0,0,0.0,null  | Each array contains the default value for its type. The default value of a primitive byte or a primitive long is printed as 0. The default value of a float primitive is printed as 0.0. The default value of an Object is null and is printed as null.  |
| 3   | e Prints: 2,5,8            | Arrays a1, a2 and a3 all contain 3 integers. The first element of the array has an index of 0 so an index of one refers to the second element of each array.   |
| 4   | a 1                        | A compile-time error occurs at the line marked 1, because the array reference declaration can not be used to declare the number of components contained in the array. Instead, the dimension expression should be contained in an array creation expression such as new int[5].  |
| 5   | e<br>Compile-time<br>error | Variable b1 is initialized to false, because it is a class member. The array component array[0] is initialized to the default value, false, of the array type, boolean[], even though the array is declared locally. Local variable b2 is not initialized, because it is local. A compile-time error is generated by the statement that attempts to print the value of b2.   |

Ask a Question  
[Send an email to me.](#)

Java Question and Answer Forums  
[JavaRanch Big Moose Saloon](#)  
[Marcus Green's Discussion Forum](#)

[java.sun.com](http://java.sun.com) Forums, Chat and User Groups

Other Resources

[Java Language Specification](#)

[Java Virtual Machine Specification](#)

[Java 2 Platform, Standard Edition, v 1.4.0 API Specification](#)

Tutorials

[Learning the Java Language](#)

[Operator Precedence Chart, Expressions, Statements, Blocks](#)

[Programming with Assertions](#)

2<sup>nd</sup> mock exam

(MULTI DIMENSION ARRAYS)

### Question 1

```
class MWC201 {  
    public static void main(String[] args) {  
        int[][] a1 = {{1,2,3},{4,5,6},{7,8,9,10}};  
        System.out.print(a1[0][2]+"," +a1[1][0]+"," +a1[2][1]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3,4,8
- b. Prints: 7,2,6
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 2

```
class MWC202 {  
    public static void main(String[] args) {  
        int[] a1[] = {{1,2},{3,4,5},{6,7,8,9}};  
        int []a2[] = {{1,2},{3,4,5},{6,7,8,9}};  
        int a3[][] = {{1,2},{3,4,5},{6,7,8,9}};  
        System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8

- e. Prints: 2,5,9
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 3

```
class MWC207 {  
    public static void main(String[] args) {  
        int[][] a1 = {{1;2},{3;4;5},{6;7;8;9}};  
        System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. Prints: 2,5,9
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 4

```
class MWC208 {  
    public static void main(String[] args) {  
        int[][] a1 = ((1,2),(3,4,5),(6,7,8,9));  
        System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. Prints: 2,5,9
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 5

```
class MWC209 {
```

```
public static void main(String[] args) {  
    int[][] a1 = [[1,2],[3,4,5],[6,7,8,9]];  
    int[][] a2 = [[1,2],[3,4,5],[6,7,8,9]];  
    int[][] a3 = [[1,2],[3,4,5],[6,7,8,9]];  
    System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. Prints: 2,5,9
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 6

```
class MWC210 {  
    public static void main(String[] args) {  
        int[] a2 = {1,2}, a3 = {3,4,5}, a4 = {6,7,8,9}; // 1  
        int[][] a1 = {a2,a3,a4}; // 2  
        System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. Prints: 2,5,9
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 7

```
class MWC211 {  
    public static void main(String[] args) {  
        int a1[3]; // 1  
        int []a2[]; // 2  
        int[ ]a3; // 3  
        int[] a4[]; // 4  
    }  
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

### Question 8

```
class MWC212 {  
    public static void main(String[] args) {  
        int[] a1[],a2[]; // 1  
        int []a3,[]a4; // 2  
        int []a5,a6[]; // 3  
        int[] a7,a8[]; // 4  
    }  
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

### Question 9

```
class MWC203 {  
    public static void main(String[] args) {  
        int[] a1[] = { new int[]{1,2},new int[]{3,4,5} };  
        int []a2[] = new int[][]{{1,2},{3,4,5}};  
        int a3[][] = {{1,2},new int[]{3,4,5}};  
        System.out.print(a1[0][1]+"," +a2[1][0]+"," +a3[1][2]);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,2,4
- d. Prints: 2,3,4
- e. Prints: 2,3,5
- f. Prints: 2,4,5
- g. Compile-time error
- h. Run-time error
- i. None of the above

### **Question 10**

```
class MWC204 {  
    public static void main(String[] args) {  
        int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};  
        System.out.print(a1.length);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0
- b. Prints: 3
- c. Prints: 4
- d. Prints: 9
- e. Prints: 10
- f. Prints: 11
- g. Compile-time error
- h. Run-time error
- i. None of the above

### **Question 11**

```
class MWC205 {  
    public static void main(String[] args) {  
        int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};  
        for (int i = 0; i < a1.length; i++) {  
            System.out.print(a1[i].length+",");  
        } } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 2,3,4,0,
- b. Prints: 1,2,5,0,
- c. Compile-time error
- d. Run-time error
- e. None of the above

### **Question 12**

```
class MWC206 {  
    public static void main (String[] args) {  
        int[][] a1 = {{1,2,3},{4,5,6},{7,8,9}};  
        for (int i = 0; i < 3; i++) {  
            for (int j = 0; j < 3; j++) {  
                System.out.print(a1[j][i]);  
            } } } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 123456789
- b. Prints: 147258369
- c. Prints: 321654987
- d. Prints: 369258147
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 13

```
class A11 {public String toString() {return "A11";}}
```

```
class A12 {
    public static void main(String[] args) {
        A11[] a1 = new A11[1];          // 1
        A11[][] a2 = new A11[2][];      // 2
        A11[][][] a3 = new A11[3][][];   // 3
        a1[0] = new A11();              // 4
        a2[0] = a2[1] = a1;            // 5
        a3[0] = a3[1] = a3[2] = a2;    // 6
        System.out.print(a3[2][1][0]);   // 7
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: null
- b. Prints: A11
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Compile-time error at 7.
- j. Run-time error
- k. None of the above

### Question 14

```
class A13 {}
class A14 {
    public static void main(String[] args) {
        A13[] a1 = new A13[1];          // 1
        A13[][] a2 = new A13[2][1];      // 2
        A13[][][] a3 = new A13[3][3][3]; // 3
        System.out.print(a3[2][2][2]);   // 4
    }
}
```

```

a1[0] = new A13();           // 5
a2[0] = a2[1] = a1;         // 6
a3[0] = a3[1] = a3[2] = a2; // 7
System.out.print(a3[2][2][2]); // 8
}

```

What is the result of attempting to compile and run the program?

- a. Prints: null
- b. Prints: nullnull
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Compile-time error at 7.
- j. Compile-time error at 8.
- k. Run-time error

#### Answers: Java Programmer Certification Mock Exam

| No. | Answer               | Remark   |
|-----|----------------------|--|
| 1   | a Prints: 3,4,8      | An array variable a1 is declared, and the declaration contains the initializer <code>{}{{1,2,3},{4,5,6},{7,8,9,10}}</code> . The initializer creates an array containing three components, and each is a reference to a subarray of type <code>int[]</code> . Each subarray contains components of type <code>int</code> , so the elements of the array referenced by a1 are of type <code>int</code> . The array access expression, <code>a1[0][2] = a1[1st subarray][third component] = 3</code> .   |
| 2   | e Prints: 2,5,9      | The declarations of the array variables, a1, a2 and a3, are different in terms of the position of the square brackets, but each is of type <code>int[][]</code> . The array access expression, <code>a1[0][1] = a[1st subarray][2nd component] = 2</code> . Each of the three array variable declarations has an array initializer. The same initializer, <code>{}{{1,2},{3,4,5},{6,7,8,9}}</code> , is used in each of the three cases. The initializer specifies three components of type <code>int[]</code> , so each component is a reference to an array object containing components of type <code>int</code> . The size of each of the subarrays is different: The size of the first is 2, the second is 3, and the third is 4. |
| 3   | f Compile-time error | The array initializer, <code>{}{{1;2};{3;4;5};{6;7;8;9}}</code> generates a compile-time error, because the commas have been replaced by semicolons. The array initializer should have been specified as follows: <code>{}{{1,2},{3,4,5},{6,7,8,9}}</code> .   |
| 4   | f Compile-time error | The array initializer, <code>((1,2),(3,4,5),(6,7,8,9))</code> , generates a compile-time error, because the curly braces have been replaced by parentheses. The array initializer should have been specified as follows: <code>{}{{1,2},{3,4,5},{6,7,8,9}}</code> .  |

|    |   |                      |   |
|----|---|----------------------|---|
| 5  | f | Compile-time error   | The array initializer, <code>[[1,2],[3,4,5],[6,7,8,9]]</code> , generates a compile-time error, because the curly braces have been replaced by square brackets. The array initializer should have been specified as follows: <code> {{1,2},{3,4,5},{6,7,8,9}} </code> .   |
| 6  | e | Prints: 2,5,9        | The line marked 1 declares three array variables, <code>a2</code> , <code>a3</code> and <code>a4</code> , and each references an array with components of type <code>int</code> . The line marked 2 declares an array variable, <code>a1</code> , where each component is initialized with a reference to one of the previously created arrays. The array access expression, <code>a1[0][1]</code> , is evaluated as follows: $a1[0][1] = a1[\text{first subarray}][\text{second component}] = 2$ .   |
| 7  | a | 1                    | A compile-time error occurs at the line marked 1, because the array variable declaration can not be used to specify the number of components contained in the array. Instead, the dimension expression should be contained in an array creation expression such as the following, new <code>int[3]</code> .   |
| 8  | b | 2                    | An array variable is declared by placing brackets after the identifier or after the type name. A compile-time error occurs at the line marked 2, because the brackets appearing before the identifier for array variable <code>a4</code> are not associated with the type or the identifier.  |
| 9  | e | Prints: 2,3,5        | Each of the three array variable declarations, <code>a1</code> , <code>a2</code> and <code>a3</code> , is different in terms of the position of the square brackets, but each declares a variable of type <code>int[][]</code> . Each of the three declarations contains an array initializer. In each case, the initializer could be simplified as follows: <code> {{1,2},{3,4,5}} </code> . The initializer creates an array containing two components, and each is a reference to an array containing components of type <code>int</code> . The size of each of the subarrays is different: The size of the first is 2 and the second is 3. The array access expression, <code>a1[0][1] = a[1st subarray][2nd component]</code> = 2. |
| 10 | c | Prints: 4            | The array initializer, <code> {{1,2},{3,4,5},{6,7,8,9},{}} </code> , creates an array containing four components, and each is a reference to a subarray of type <code>int[]</code> . The size of the array referenced by variable <code>a1</code> , is 4, because <code>a1</code> contains four components. The size of the first subarray is 2, the second is 3, the third is 4 and the fourth is zero.  |
| 11 | a | Prints:<br>2,3,4,0,  | The array initializer, <code> {{1,2},{3,4,5},{6,7,8,9},{}} </code> , creates an array containing four components, and each is a reference to an array of type <code>int[]</code> . The size of the first subarray is 2, the second is 3, the third is 4, and the fourth is zero. While the components of the array referenced by <code>a1</code> are of type <code>int[]</code> , the elements of the array referenced by <code>a1</code> are of type <code>int</code> .  |
| 12 | b | Prints:<br>147258369 | The array variable <code>a1</code> is declared with the initializer, <code> {{1,2,3},{4,5,6},{7,8,9}} </code> . The array access expression, <code>a1[0][1] = a1[\text{first subarray}][\text{second element}] = 2. If the argument of the print statement had been <code>a1[i][j]</code> then the output would have been 123456789. The tricky feature of this question is the reversal of i and j to produce the deceptive array access expression, <code>a1[j][i]</code>. The</code>   |

|    |                                    |  |
|----|------------------------------------|--|
|    |                                    | output is 147258369.   |
| 13 | b Prints: A11                      | <p>The declaration <code>A11[] a1 = new A11[1]</code> declares a variable <code>a1</code> that references an array that contains one component of type <code>A11</code>. The declaration <code>A11[][] a2 = new A11[2][]</code> declares a variable <code>a2</code> that references an array that contains two components of type <code>A11[]</code>. In other words, the array referenced by <code>a2</code> contains two reference variables, and each is able to reference a subarray. The initial value of the subarray references is null. The size of the subarrays has not been specified. The declaration <code>A11[][][] a3 = new A11[3][][]</code> declares a variable <code>a3</code> that references an array that contains three components of type <code>A11[][]</code>. In other words, the array referenced by <code>a3</code> contains three reference variables, and each is able to reference a subarray. The initial value of each subarray reference is null. The dimensions of the subarrays have not been specified. At line 5, a reference to the array referenced by <code>a1</code> is assigned to each of the two components of the array referenced by <code>a2</code>, <code>a2[0] = a2[1] = a1</code>. At line 6, a reference to the array referenced by <code>a2</code> is assigned to each of the three components of the array referenced by <code>a3</code>, <code>a3[0] = a3[1] = a3[2] = a2</code>. In other words, after line 6, each component of the array referenced by <code>a3</code> is a reference to the array referenced by <code>a2</code>, and each element of the array referenced by <code>a2</code> is a reference to the array referenced by <code>a1</code>, and the array referenced by <code>a1</code> contains a reference to an instance of class <code>A11</code>. Every element of the multi-dimensional array referenced by <code>a3</code> contains a reference to a single instance of class <code>A11</code>. The print method invokes the <code>toString</code> method on the instance, and produces the output, <code>A11</code>.</p> |
| 14 | a Prints: null<br>k Run-time error | <p>The declaration <code>A13[] a1 = new A13[1]</code> declares a variable <code>a1</code> that references an array that contains one component of type <code>A13</code>. The declaration <code>A13[][] a2 = new A13[2][1]</code> declares a variable <code>a2</code> that references an array that contains two components of type <code>A13[]</code>. In other words, the array referenced by <code>a2</code> contains two references to two subarrays of type <code>A13[]</code>. The size of each subarray is 1. The declaration <code>A13[][][] a3 = new A13[3][3][3]</code> declares a variable <code>a3</code> that references an array that contains three components of type <code>A13[][]</code>. In other words, the array referenced by <code>a3</code> contains three references to three subarrays of type <code>A13[][]</code>. The dimensions of the subarrays are <math>3 \times 3</math>. The dimensions of the array referenced by <code>a3</code> are <math>3 \times 3 \times 3</math>. The number of elements in the array referenced by <code>a3</code> is <math>3 * 3 * 3 = 27</math> elements. Each of the three subarrays contains three components, where each is a reference to a subarray of type <code>A13[]</code>. Each of the 27 elements of the array referenced by <code>a3</code> is a reference of type <code>A13</code> that has been initialized to the default value of null. At line 7, a reference to the array referenced by <code>a2</code> is assigned to each of the three components of the array referenced by <code>a3</code>, <code>a3[0] = a3[1] = a3[2] = a2</code>. Since the array referenced by <code>a2</code> has dimensions <math>2 \times 1</math>, the array referenced by <code>a3</code> now has dimensions <math>3 \times 3</math>.</p>   |

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

X 2 X 1. The number of elements is 6. An attempt to access any element beyond those 6 results in an exception at run-time.

Mock exam3

## CLASS DECLARATION

### Question 1

```
public class Basics {} // 1
class Basics1 {} // 2
protected class Basics2 {} // 3
private class Basics3 {} // 4
Class Basics4 {} // 5
```

Suppose these are top-level class declarations and not nested class declarations; and suppose that all of the declarations are contained in one file named Basics.java. Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

### Question 2

```
public class Basics {} // 1
public class Basics2 {} // 2
public class Basics3 {} // 3
public class Basics4 {} // 4
```

Suppose these are top-level class declarations and not nested class declarations; and suppose that all of the declarations are contained in one file named Basics.java. A compile-time error is not generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

### Question 3

Which of the following modifiers can be applied to a class that is not a nested class?

- a. public
- b. protected
- c. private
- d. abstract
- e. static
- f. final

#### Question 4

Which of the follow statements is true.

- a. An anonymous class can be declared abstract.
- b. A local class can be declared abstract.
- c. An abstract class can be instantiated.
- d. An abstract class is implicitly final.
- e. An abstract class must declare at least one abstract method.
- f. An abstract class can not extend a concrete class.

#### Question 5

```
public class A {int i1; void m1() {}}
```

Which of the following statements are true?

- a. class A extends Object.
- b. Field i1 is implicitly public, because class A is public.
- c. Method m1 is implicitly public, because class A is public.
- d. The compiler will insert a default constructor implicitly.
- e. The default constructor has no throws clause.
- f. The default constructor of class A has package access.
- g. The default constructor accepts one parameter for each field in class A.
- h. The default constructor invokes the no-parameter constructor of the superclass.

#### Question 6

```
abstract class A {} // 1  
transient class G {} // 2  
private class C {} // 3  
static class E {} // 4
```

Suppose these are top-level class declarations and not nested class declarations. Which of these declarations would not produce a compile-time error?

- a. 1
- b. 2

- c. 3
- d. 4
- e. None of the above

### Question 7

```
protected class D {} // 1
synchronized class F {} // 2
volatile class H {} // 3
final class B {} // 4
```

Suppose these are top-level class declarations and not nested class declarations. Which of these declarations would not produce a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

| No. | Answer                       | Remark  |
|-----|------------------------------|---|
| 1   | c<br>d 3 4 5<br>e            | If a class C is declared as a member of an enclosing class then C may be declared using no access modifier or any of the three access modifiers, private, protected or public. However, if class C is not a local class, anonymous class or a member of an enclosing class or interface; then C may be declared with the public modifier or with package access (i.e. no modifier). The other two access modifiers, private and protected, are not applicable to any class that is not a member class. The class declaration, Class Basics4 {}, generates a compile-time error, because all of the letters of the reserved word class must be lower case. |
| 2   | a 1                          | Only one class in a source code file can be declared public. The other classes may not be public. Therefore, the declarations for classes Basics2, Basics3 and Basics4 generate compile-time errors.  |
| 3   | a<br>d public abstract final | The access modifiers, protected and private, can be applied to a class that is a member of an enclosing class, but can not be applied to a local class or a class that is not nested inside another class. The static modifier can be applied to a class that is a member of an enclosing class, but  |

|   |                  |  |  |
|---|------------------|--|--|
|   |                  |  | can not be applied to a local class or a class that is not nested inside another class. The public modifier can be applied to a top level class to allow the class to be accessed from outside of the package. The abstract modifier prevents the class from being instantiated. An abstract class may include zero, one or more abstract methods. The final modifier prevents a class from being extended.      |
| 4 | b                | A local class can be declared abstract.  | An anonymous class can not be extended; therefore, an anonymous class can not be declared abstract. A local class can be abstract. An abstract class can not be instantiated. If a class declaration contains an abstract method, then the class must also be declared abstract. A class can be declared abstract even if it does not contain an abstract method. An abstract class can never be declared final. |
| 5 | a<br>d<br>e<br>h | class A extends Object. The compiler will insert a default constructor implicitly. The default constructor has no throws clause. The default constructor invokes the no-parameter constructor of the superclass. | Field i1 and m1 both have package access. When no constructor is declared explicitly the compiler will insert one implicitly. The implicitly declared default constructor will have the same access privileges as the class. In this case, the class is public, so the default constructor is also public. The default constructor accepts no parameters and throws no exceptions.                               |
| 6 | a<br>l           |  | The modifiers, private and static, can be applied to a nested class, but can not be applied to a class that is not nested. A class that is not nested can have public or package access, but not private. The transient modifier can not be applied to any class; because it is a field modifier.  |
| 7 | d<br>4           |  | The modifier, protected, can not be applied to a top level class, but can be applied to a nested class. The synchronized modifier can not be applied to any class, because it is a method modifier. The modifier, volatile, can not be applied to any class; because it is a field modifier.   |

## MOCKEXAM4

### NESTED CLASS DECLARATION

#### Question 1

Which of the follow are true statements.

- a. A nested class is any class that is declared within the body of another class or interface.
- b. A nested class can not be declared within the body of an interface declaration.
- c. An inner class is a nested class that is not static.
- d. A nested class can not be declared static.
- e. A named class is any class that is not anonymous.

### Question 2

Which of the follow are true statements.

- a. A local class is declared within a method, constructor or block.
- b. An anonymous class is always a local class.
- c. A local class is a nested class.
- d. A local class is a member class.
- e. A local class is always a named class.

### Question 3

Which of the following are class modifiers? Select all that are applicable to a top-level class and all that are applicable to a nested class. It is not required that a modifier be applicable to both.

- a. abstract
- b. extends
- c. final
- d. implements
- e. private
- f. protected
- g. public
- h. static
- i. synchronized
- j. transient
- k. volatile

### Question 4

Which of the following modifiers can be applied to a member class?

- a. abstract
- b. final
- c. public
- d. protected
- e. private

- f. static
- g. synchronized
- h. transient

### Question 5

Which of the following modifiers can be applied to a local class?

- a. public
- b. protected
- c. private
- d. abstract
- e. static
- f. final

### Question 6

```
class Z {  
    abstract class A {}    // 1  
    final class B {}      // 2  
    private class C {}    // 3  
    protected class D {}  // 4  
    public class E {}     // 5  
}
```

Which class declaration results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. None of the above

### Question 7

```
class Z {  
    static class F {}      // 1  
    synchronized class G {} // 2  
    transient class H {}   // 3  
    volatile class I {}    // 4  
}
```

Which class declaration does not result in a compile-time error?

- a. 1
- b. 2

- c. 3
- d. 4
- e. None of the above

### Question 8

```
class Z {  
    void m1() {  
        abstract class A {} // 1  
        final class B {} // 2  
        private class C {} // 3  
        protected class D {} // 4  
        public class E {} // 5  
    }  
}
```

Which class declarations result in compile-time errors?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

### Question 9

```
class Z {  
    void m1() {  
        static class F {} // 1  
        synchronized class G {} // 2  
        transient class H {} // 3  
        volatile class I {} // 4  
        abstract class A {} // 5  
        final class B {} // 6  
    }  
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6

### **Question 10**

```
class A {  
    A() {}      // 1  
    int A;      // 2  
    void A() {} // 3  
    class A {} // 4  
}
```

Which line results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

### **Question 11**

```
class A {  
    int B;      // 1  
    void B() {} // 2  
    class B {} // 3  
}
```

Which line results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. None of the above

### **Question 12**

```
abstract class A {}           // 1  
private abstract void m1(); // 2  
private abstract class B {} // 3  
private class C extends B {} // 4  
}
```

Which line results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above.

### Question 13

```
abstract class A {           // 1
    abstract final void m1(); // 2
    abstract final class B {} // 3
    class C extends B {}     // 4
}
```

Which line does not result in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

### Question 14

```
abstract class A {           // 1
    abstract synchronized void m1(); // 2
    abstract synchronized class B {} // 3
    synchronized class C extends B {} // 4
}
```

Which line does not result in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

| No. | Answer   | Remark  |
|-----|--|---|
| 1   | A nested class is any class that is declared within the body of another class or interface. An inner class is a nested class that is not static. A named class is any class that is not anonymous. | Every class declared within the body of another class or interface is known as a nested class. If the nested class does not have a name, then it is an anonymous class. If the nested class has a name, then it is not anonymous. If the nested class has a name and is not declared inside of a method, constructor or any block, then it is a member class. If a member class is not static, then it is an inner class. If a class is not nested, then it is a top level class. A nested class that is static is sometimes referred to as a top level class, but that usage of the term is confusing and should be avoided. |
| 2   | a A local class is declared within a method, constructor or  | Every class declared within the body of another class is known as a nested class. If the nested   |

|   |                            |  |   |
|---|----------------------------|--|---|
|   | e                          | block. A local class is a nested class. A local class is always a named class. | class does not have a name, then it is an anonymous class. If the nested class has a name, then it is not anonymous. If the nested class has a name and is declared inside of a method, constructor or any block, then it is a local class. If a nested class does not have a name, then it can not be called a local class even if it is declared inside of a block. Therefore, an anonymous class is never called a local class. If the nested class has a name and is not declared inside of a method, constructor or any block, then it is a member class. If the class is not nested, then it is a top level class. Please note that this question is more rigorous than those that one might expect to find on the real exam. It has been included here as a convenient place to define some terms that are used to explain the answers to some of the other questions that appear in this mock exam. |
| 3 | a<br>c<br>e<br>f<br>g<br>h | abstract final private<br>protected public static                              | The access modifiers, public, protected and private, can not be applied to a local class. The protected and private access modifiers can be applied to member classes, but not to a top level class. The public modifier can be applied to a top level class or a member class, but not a local class. The static modifier can be applied to a member class, but not to a local class or top level class. The keywords extends and implements are not modifiers. The synchronized modifier can be applied to a method, but not to a class. The modifiers, transient and volatile, can be applied to a field, but not to a class.  |
| 4 | a<br>b<br>c<br>d<br>e<br>f | abstract final public<br>protected private static                              | A nested class that has a name and is not a local class is a member class. A member class can be static or non-static. A non-static member class is also known as an inner class. All of the class modifiers may be applied to a member class. The modifiers, synchronized and transient, are not class modifiers.  |
| 5 | d<br>f                     | abstract final   | If a nested class has a name and is declared inside of a method, constructor or any block, then it is a local class. No access modifier can be applied to a local class declaration. A local class can not be static. A local class can be abstract, and can be final.  |
| 6 | f                          | None of the above  | The following modifiers can be applied to a member class: abstract, private, protected, public,   |

|    |                          |  |   |
|----|--------------------------|--|---|
|    |                          |  | static and final.   |
| 7  | a 1                      |  | Please note that this question asks which class declaration does NOT result in a compile-time error. The following modifiers can be applied to a member class: abstract, private, protected, public, static and final. The synchronized modifier can not be applied to any class, because it is a method modifier. The modifiers, transient and volatile, can not be applied to any class, because they are field modifiers.  |
| 8  | c<br>d<br>e 3 4 5        |  | The abstract and final modifiers can be applied to a method local class declaration.  |
| 9  | a<br>b<br>c<br>d 1 2 3 4 |  | The modifiers, abstract and final, can be applied to a method local class declaration. The synchronized modifier can not be applied to a class, because it is a method modifier. The modifiers, transient and volatile, can not be applied to any class, because they are field modifiers.  |
| 10 | d 4                      |  | A method and a field can share the same name, because they are used in different contexts and use different lookup procedures. They can even share the same name with the class in which they are declared. Please note that class names usually begin with an upper case letter while method and field names usually begin with a lower case letter. A nested class can not share the same name with its enclosing class.  |
| 11 | d None of the above      |  | A method, field, and a nested class can share the same name, because they are used in different contexts and use different lookup procedures. Please note that class names usually begin with an upper case letter while method and field names usually begin with a lower case letter. Also note that a nested class can not share the same name with its enclosing class; however, a method and field can share a name with the enclosing class. Even so, it is not a good idea to name a method with the name of the enclosing class, because it could be confused with a constructor. |
| 12 | b 2                      |  | An abstract method has no implementation, and is not useful until an extending class implements the method. Private methods are not inherited and can not be overridden. If an abstract method is   |

|    |   |   |   |
|----|---|---|---|
|    |   |   | declared private, then it can not be implemented in a subclass. Although a method may not be both private and abstract, a nested class can be; because another nested class can extend the abstract class and implement any abstract methods.   |
| 13 | a | 1 | Please note that this question asks which line does NOT result in a compile-time error. An abstract method has no implementation and is not useful until an extending class implements the method. A final method can not be overridden by a subclass method. An abstract final method can not be implemented and is not legal. An abstract class may contain abstract method declarations and is assumed to be incomplete. A final class can not be extended. The implementation of an abstract final class could not be completed. The declaration of class C results in a compiler error, because a final class may not be listed in the extends clause. |
| 14 | a | 1 | Please note that this question asks which line does NOT result in a compile-time error. The modifier, synchronized, is a method modifier, but is not a class modifier. Any attempt to declare a synchronized class results in a compile-time error. Since the synchronized modifier specifies an implementation detail it makes no sense to use it with an abstract method that provides no implementation.   |

## MOCKEXAMS ANNONYMUS CLASSES

### Question 1

Which of the following is a true statement?

- a. An anonymous class can extend only the Object class.
- b. An anonymous class can not implement an interface.
- c. An anonymous class declaration can not have an implements clause.
- d. An anonymous class declaration can name more than one interface in the implements clause.
- e. The class instance creation expression for an anonymous class must never include arguments.
- f. None of the above

## Question 2

Which of the following are true statements?

- a. An anonymous class is implicitly abstract.
- b. An anonymous class is implicitly final.
- c. An anonymous class is implicitly static.
- d. A static reference variable can reference an instance of an anonymous class.
- e. An anonymous class declaration must have at least one explicit constructor declaration.
- f. An anonymous class declaration can have more than one explicit constructor declaration.

## Question 3

```
abstract class A {  
    private int x = 4, y = 2;  
    public int x() {return x;}  
    public void x(int x) {this.x = x;}  
    public int y() {return y;}  
    public void y(int y) {this.y = y;}  
    public abstract int math();  
}  
class B {  
    static A a1 = new A(2,1) {  
        public A(int i1, int i2) {x(i1);y(i2);};  
        public int math() {return x() + y();}  
    };  
    public static void main(String[] args) {  
        System.out.print(a1.math());  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 8
- b. Prints: 3122
- c. Compile-time error
- d. Run-time error
- e. None of the above

## Question 4

```
class A {  
    private static int f1 = 1;  
    private int f2 = 2;
```

```

void m1(int p1, final int p2) {
    int l1 = 5;
    final int l2 = 6;
    Object x = new Object() {
        int a = f1; // 1
        int b = f2; // 2
        int c = p1; // 3
        int d = p2; // 4
        int e = l1; // 5
        int f = l2; // 6
    };}}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6

### Question 5

```

abstract class A {
    private int x = 1, y = 1;
    public A(int x, int y) {this.x = x; this.y = y;}
    public abstract int math();
}
class B {
    static A a1 = new A(2,1) {public int math() {return x + y; }};
    static A a2 = new A(2,1) {public int math() {return x - y; }};
    static A a3 = new A(2,1) {public int math() {return x * y; }};
    static A a4 = new A(2,1) {public int math() {return x / y; }};
    public static void main(String[] args) {
        System.out.print(" " + a1.math() + a2.math() +
                        a3.math() + a4.math());
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3122
- b. Prints: 2011
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 6

```

abstract class A {
    private int x = 4, y = 2;
    public int x() {return x;}
    public void x(int x) {this.x = x;}
    public int y() {return y;}
    public void y(int y) {this.y = y;}
    public abstract int math();
}
class B {
    static A a1 = new A() {public int math() {return x() + y();}}
    static A a2 = new A() {public int math() {return x() - y();}}
    static A a3 = new A() {public int math() {return x() * y();}}
    static A a4 = new A() {public int math() {return x() / y();}}
    public static void main(String[] args) {
        System.out.print(" " + a1.math() + a2.math() +
                        a3.math() + a4.math());
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 18
- b. Prints: 6282
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 7

```

class A {String m1() {return "A.m1";}}
interface B {String m2();}
class C {
    static class D extends A implements B {
        public String m1() {return "D.m1";}
        public String m2() {return "D.m2";}
    }
    static A a1 = new A() implements B {
        public String m1() {return "m1";}
        public String m2() {return "m2";}
    };
    public static void main(String[] args) {
        System.out.print(a1.m1() + "," + new C.D().m2());
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: m1,D.m2

- b. Prints: A.m1,D.m2
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 8

```

abstract class A {
    private int x = 4, y = 2;
    public A(int i1, int i2) {x=i1;y=i2;}
    public int x() {return x;}
    public void x(int x) {this.x = x;}
    public int y() {return y;}
    public void y(int y) {this.y = y;}
    public abstract int math();
}
class B {
    static A a1 = new A(2,1) {public int math() {return x() + y();}};
    static A a2 = new A(2,1) {public int math() {return x() - y();}};
    static A a3 = new A(2,1) {public int math() {return x() * y();}};
    static A a4 = new A(2,1) {public int math() {return x() / y();}};
    public static void main(String[] args) {
        System.out.print(" " + a1.math() + a2.math() +
                        a3.math() + a4.math());
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 8
- b. Prints: 3122
- c. Compile-time error
- d. Run-time error
- e. None of the above

| No. | Answer   | Remark   |
|-----|--|--|
| 1   | c<br>An anonymous class declaration can not have an implements clause. | A class instance creation expression can create an instance of a named class or an anonymous class. For example, the class instance creation expression <code>new Object()</code> creates an instance of the class named <code>Object</code> . If a class body appears in the class instance creation expression, then an anonymous class is created. For example, the expression <code>new Object() { void doNothing(){} }</code> creates an instance of an anonymous class that extends <code>Object</code> and implements a method named <code>doNothing</code> . In other words, if a class name immediately follows the keyword <code>new</code> , then the |

|   |     |  |  |
|---|-----|--|--|
|   |     |  | anonymous class extends the named class. When a named class is being extended, then the class instance creation expression can contain an optional argument list. The arguments will be passed to the direct superclass constructor that has a matching parameter list. An anonymous class declaration can not have an implements clause or an extends clause.   |
| 2 | b d | An anonymous class is implicitly final. A static reference variable can reference an instance of an anonymous class. | An anonymous class can extend Object and implement an interface, or the anonymous class can extend a named class including Object. An anonymous class can not be extended; so it can not be abstract. An anonymous class declaration always creates an instance of a class; so it is not surprising that an anonymous class can not be declared static. Even so, a static reference variable can refer to an anonymous class. A constructor shares the same name as the class in which it is declared, but an anonymous class has no name. For that reason, it is not surprising that an anonymous class declaration can not contain an explicit constructor declaration. Instead, an anonymous class can contain an instance initializer. |
| 3 | c   | Compile-time error   | An anonymous class declaration can not contain an explicit declaration of a constructor.   |
| 4 | c e | 3 5  | Local method variables and method parameters are stored on the stack and go out of scope after the method is exited. Although a local reference variable is stored on the stack, the referenced object is stored on the heap; so the object can continue to exist long after the method runs to completion. An object that is instantiated within a method or block is not permitted to refer to a variable that is declared within the method or block unless the variable is declared final and the variable declaration precedes the creation of the object.  |
| 5 | c   | Compile-time error   | The instance variables x and y in class A are private, so they are not accessible to the anonymous subclasses. If you want to access the private variables of a superclass, then you will need to add accessor methods to the superclass such as getX and getY.  |
| 6 | c   | Compile-time error   | When an anonymous class declaration is the last thing that appears in a statement, then a semicolon  |

|   |   |                    |   |
|---|---|--------------------|---|
|   |   |                    | must follow the declaration. Anonymous class declarations provide an excellent opportunity for trick questions involving statements with missing semicolons.  |
| 7 | c | Compile-time error | An anonymous class can extend Object and implement an interface, or the anonymous class can extend a named class including Object. An anonymous class declaration can not have an implements clause. In this case, the declaration of the anonymous class referenced by a1 generates a compile-time error as a result of the attempt to extend class A and implement interface B. |
| 8 | b | Prints: 3122       | The arguments that appear in the class instance creation expression of an anonymous class are passed to a constructor of the superclass.  |

## MOCKEXAM6 CONSTRUCTORS

### Question 1

```
class A {A(int i) {} } // 1
class B extends A {} // 2
```

Which of the following statements are true?

- a. The compiler attempts to create a default constructor for class A.
- b. The compiler attempts to create a default constructor for class B.
- c. Compile-time error at 1.
- d. Compile-time error at 2.

### Question 2

Which of the following modifiers can be applied to a constructor?

- a. private
- b. abstract
- c. final
- d. volatile
- e. native
- f. None of the above.

### Question 3

Which of the following techniques can be used to prevent the instantiation of a class by any code outside of the class?

- a. Do not declare any constructors.
- b. Do not use a return statement in the constructor.
- c. Declare all constructors using the keyword void to indicate that nothing is returned.
- d. Declare all constructors using the private access modifier.
- e. None of the above.

#### Question 4

Which of the following modifiers can be applied to a constructor?

- a. protected
- b. public
- c. static
- d. synchronized
- e. transient

#### Question 5

Which of the following statements are true?

- a. The compiler will create a default constructor if no other constructor is declared.
- b. The default constructor takes no arguments.
- c. If a class A has a direct superclass, then the default constructor of class A invokes the no-argument constructor of the superclass.
- d. The default constructor declares Exception in the throws clause.
- e. The default constructor is always given the private access modifier.
- f. The default constructor is always given the public modifier.
- g. The default constructor is always given default package access.

#### Question 6

Suppose that the compiler generates a default constructor for a class. If a compile-time error is to be avoided, which of the following must be true?

- a. The superclass must not have any constructor other than a default constructor.
- b. The superclass must not have an accessible no-argument constructor.
- c. The no-argument superclass constructor must not have a throws clause that includes a checked exception.
- d. The no-argument superclass constructor must be declared private.
- e. None of the above

#### Question 7

```

class A {A() throws Exception {}} // 1
class B extends A {B() throws Exception {}} // 2
class C extends A {} // 3

```

Which of the following statements is true?

- Compile-time error at 1.
- Compile-time error at 2.
- Compile-time error at 3.
- None of the above

| No. | Answer  | Remark   |
|-----|---|--|
| 1   | b<br>d<br>The compiler attempts to create a default constructor for class B. Compile-time error at 2. | If no constructor is declared explicitly, then the compiler will implicitly create a default constructor that accepts no parameters, has no throws clause, and invokes its superclass constructor. Since class A has an explicitly declared constructor, the compiler will not create an implicit default constructor. Class B does not have an explicit constructor declaration, so the compiler attempts to create a default constructor. Since class A does not have a no-parameter constructor, the attempt by class B to invoke the no parameter constructor of A would fail. As a result, a compiler error is generated at marker 2. |
| 2   | a private   | Constructors are not inherited and can not be overridden, so there is no need for the final modifier in a constructor declaration. Furthermore, an abstract constructor would be useless, since it could never be implemented. The volatile modifier can be applied to a field, but not to a constructor. Native constructors are not permitted, because it would be difficult for Java to verify that the native constructor properly invokes the superclass constructor.   |
| 3   | d<br>Declare all constructors using the private access modifier.                                      | If no constructors are declared explicitly; then the compiler will create one implicitly, and it will have the same access modifier as the class. The explicit declaration of any constructor will prevent the creation of a default constructor. If all constructors are declared private, then code outside of the class will not have access to the constructors and will not have the ability to create an instance of the class. Constructors do not  |

|   |             |  |  |
|---|-------------|--|--|
|   |             |  | return a value and constructor declarations do not include a return type, so the keyword void is not applicable to a constructor declaration.  |
| 4 | a<br>b      | protected public   | Constructors can not be inherited, so an abstract constructor would be useless, since it could never be implemented. A static constructor would also be useless--or nearly so--since it would be unable to access the non-static members of the new instance. An object is not available to multiple threads during the construction process, so the synchronized modifier would not provide additional protection. The transient modifier can be applied to a field, but not a constructor.   |
| 5 | a<br>b<br>c | The compiler will create a default constructor if no other constructor is declared. The default constructor takes no arguments. If a class A has a direct superclass, then the default constructor of class A invokes the no-argument constructor of the superclass. | If no constructor is declared explicitly, then the compiler will implicitly insert a default constructor. The default constructor takes no arguments. The primordial class Object has no superclass; so the default constructor of type Object does not invoke a superclass constructor. If a class A has a direct superclass, then the default constructor of class A will invoke the no-argument superclass constructor. It is unlikely that the real exam would try to trick you with a question that requires you to know that the constructor of type Object does not invoke a superclass constructor. For the purposes of the real exam, it might be safer to overlook that particular unique feature of type Object. If a subclass constructor attempts to invoke the no-argument superclass constructor when none exists, then a compile-time error is generated. The access modifier implicitly assigned to the default constructor is the same as that assigned to the class. The default constructor does not have a throws clause. Consequently, a compile-time error is generated if the no-argument constructor of the superclass has a throws clause. |
| 6 | c           | The no-argument superclass constructor must not have a throws clause that includes a checked exception.  | The default constructor takes no arguments, and it invokes the superclass constructor with no arguments. If the superclass does not have an accessible no-argument constructor, then a compile-time error is generated. The default  |

|   |   |                          |  |
|---|---|--------------------------|--|
|   |   |                          | constructor does not have a throws clause. Consequently, a compile-time error is generated if the no-parameter constructor of the superclass has a throws clause.  |
| 7 | c | Compile-time error at 3. | The compiler creates a constructor for class C implicitly. The implicitly created constructor accepts no parameters and has no throws clause. The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A implicitly, both B and C must declare Exception in their throws clause. A compile-time error is generated at marker 3, because the default constructor does not declare Exception in the throws clause. |

## MOCKEXAM7 FIELD DECLARATIONS

### Question 1

```
class JSC102 {
    public static void main (String[] args) {
        private int x = 1; protected int y = 2; public int z = 3;
        System.out.println(x+y+z);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 123
- b. Prints: 1 2 3
- c. Prints: 6
- d. Run-time error
- e. Compile-time error
- f. None of the above

### Question 2

```
class JSC105 {
    private static int x; protected static int y; public static int z;
    public static void main (String[] args) {System.out.println(x+y+z);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing.
- b. Prints an undefined value.
- c. Prints: null
- d. Prints: 0
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 3

```
class Red {  
    public int a; public static int b;  
    public static void main (String[] in) {  
        Red r1 = new Red(), r2 = new Red(); r1.a++; r1.b++;  
        System.out.print(r1.a+, "+r1.b+, "+r2.a+, "+r2.b);  
    }}  
What is the result of attempting to compile and run the program?
```

- a. Prints: 0, 0, 0, 0
- b. Prints: 0, 1, 1, 1
- c. Prints: 1, 1, 1, 0
- d. Prints: 1, 1, 0, 1
- e. Prints: 1, 1, 0, 0
- f. Prints: 1, 1, 1, 1
- g. Compile-time error
- h. Run-time error
- i. None of the above

### Question 4

```
class Basics {  
    int x = 1, y = 2;  
    public static void main (String[] args) {System.out.println(x+y);}  
}  
What is the result of attempting to compile and run the program?
```

- a. Prints: x+y
- b. Prints: 12
- c. Prints: 3
- d. Run-time error
- e. Compile-time error
- f. None of the above

### Question 5

Which of the following modifiers can be applied to the declaration of a field?

- a. abstract
- b. final
- c. private
- d. protected
- e. public

#### **Question 6**

Which of the following modifiers can be applied to the declaration of a field?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. native

#### **Question 7**

Which of the following is used to prevent the serialization of a non-static field?

- a. final
- b. protected
- c. synchronized
- d. transient
- e. volatile
- f. native

#### **Question 8**

Which of the following statements are true?

- a. A value can not be assigned to a final field more than once.
- b. A value can be assigned to a final field at any time or not at all.
- c. Only static variables can be declared final.
- d. A compile-time error is thrown if a blank final instance variable is not assigned a value before the end of each constructor.
- e. A field can not be declared both final and volatile.

#### **Question 9**

```
class JSC101 {  
    void m1() {  
        public int a; // 1
```

```
protected int b; // 2
private int c; // 3
static int d; // 4
transient int e; // 5
volatile int f; // 6
final int g = 1; // 7
}}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6
- g. 7

#### Question 10

```
class JSC103 {
    transient float e = 1; // 1
    volatile float f = 1; // 2
    abstract float j = 1; // 3
    final float g = 1; // 4
    private final float k = 1; // 5
    private transient float l = 1; // 6
    volatile final float m = 1; // 7
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6
- g. 7

#### Question 11

```
class JSC104{
    public float a; // 1
    protected float b; // 2
    private float c; // 3
    static float d; // 4
```

```
synchronized float i; // 5  
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. None of the above.

### Question 12

```
class Identifiers {  
    int i1; // 1  
    int _i2; // 2  
    int i_3; // 3  
    int #i4; // 4  
    int $i5; // 5  
    int %i6; // 6  
    int i$7; // 7  
    int 8i; // 8  
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6
- g. 7
- h. 8

| No. | Answer                  | Remark   |
|-----|-------------------------|--|
| 1   | e<br>Compile-time error | The access modifiers public, protected and private, can not be applied to variables declared inside methods.   |
| 2   | d<br>Prints: 0          | Member variables are initialized automatically. Type int variables are initialized to zero.  |
| 3   | d<br>Prints: 1, 1, 0, 1 | Both instances of class Red share a single copy of the static field b. Although field b is only incremented using the r1 reference, the change is visible in the r2 instance of class Red. |

|   |                  |   |   |
|---|------------------|---|---|
| 4 | e                | Compile-time error  | A static method can not access a non-static variable.   |
| 5 | b<br>c<br>d<br>e | final private protected public  | A field is a class member. A static field is sometimes called a class variable. A non-static field is sometimes called an instance variable. A variable declaration that is immediately contained by a block such as a method body is called a local variable. The access modifiers, private, protected and public, can be applied to a field. A final field can not have its value assigned more than once. The abstract modifier may be applied to methods but not to fields.   |
| 6 | a<br>c<br>d      | static transient volatile   | A transient field is not part of the persistent state of an object. Transient fields are not serialized. Fields that are shared between threads may be marked volatile to force each thread to reconcile its own working copy of the field with the master copy stored in the main memory. The synchronized modifier may be applied to methods but not to fields.   |
| 7 | d                | transient   | A transient field is not part of the persistent state of an object, so it is not serialized. A static field is also not part of the persistent state of an object, and also is not serialized.  |
| 8 | a<br>d<br>e      | A value can not be assigned to a final field more than once. A compile-time error is thrown if a blank final instance variable is not assigned a value before the end of each constructor. A field can not be declared both final and volatile. | Static and non-static field variables may be declared final. All final fields must be definitely assigned a value once and only once. If the declaration of a final variable does not include an initializer then the variable is called a blank final. All blank, final, static variables must be assigned in a static initializer. All blank final non-static variables must be assigned by the end of the instance construction process. A field is sometimes shared between threads. The volatile modifier is used to force threads to reconcile their own working copy of a field with the master copy in main memory. If a field is declared final then its value does not change and there is no need for threads to reconcile their own working copies of the variable with the master copy in main memory. |
| 9 | a<br>b           | 1 2 3 4 5 6   | A variable that is local to a method can not be accessed from outside of the class, so the  |

|    |                   |   |
|----|-------------------|---|
|    | c<br>d<br>e<br>f  | access modifiers are not useful and not legal. A variable that is local to a method can not be part of the persistent state of an object, so the transient modifier is not useful and not legal. Local variables can not be shared between threads, so the volatile modifier is not useful and not legal. A local variable can be declared final to prevent its value from being assigned more than once. If the value of the variable needs to be accessed from a local class or an anonymous class, then the local variable or method parameter must be declared final. |
| 10 | c<br>g 3 7        | The abstract modifier can be applied to a class and a method but not a field. A field is sometimes shared between threads. The volatile modifier is used to force threads to reconcile their own working copy of a field with the master copy in main memory. If a field is declared final then its value does not change and there is no need for threads to reconcile their own working copies of the variable with the master copy in main memory.   |
| 11 | e 5               | The synchronized modifier is a method modifier and can not be applied to a field.   |
| 12 | d<br>f 4 6 8<br>h | The first letter of an identifier can be any <i>Unicode JLS 3.1</i> character that is a <i>Java letter</i> . The first letter can not be a number. For historical reasons, the dollar sign \$ and underscore _ are considered Java letters along with many currency symbols in use in the world today.  |

## MOCKEXAM8 METHOD DECLARATIONS

### Question 1

Which of the following modifiers can not be applied to a method?

- a. abstract
- b. private
- c. protected
- d. public

- e. volatile
- f. None of the above.

### Question 2

Which of the following modifiers can not be applied to a method?

- a. final
- b. static
- c. synchronized
- d. transient
- e. native
- f. None of the above.

### Question 3

Which of the following modifiers can not be used with the abstract modifier in a method declaration?

- a. final
- b. private
- c. protected
- d. public
- e. static
- f. synchronized
- g. native

### Question 4

Which of the following statements is true?

- a. An abstract method can not be overridden by an abstract method.
- b. An instance method that is not abstract can not be overridden by an abstract method.
- c. An abstract method declaration can not include a throws clause.
- d. The body of an abstract method is represented by a set of empty brackets.
- e. None of the above.

### Question 5

Which of the following statements is not true?

- a. A static method is also known as a class method.
- b. A class method is not associated with a particular instance of the class.
- c. The keyword, this, can not be used inside the body of a static method.
- d. The keyword, super, may be used in the body of a static method.

- e. A method that is not static is known as an instance method.
- f. None of the above.

### Question 6

Which of the following statements are true?

- a. A final method can not be overridden.
- b. All methods declared in a final class are implicitly final.
- c. The methods declared in a final class must be explicitly declared final or a compile-time error occurs.
- d. It is a compile-time error if a private method is declared final.
- e. A machine-code generator can inline the body of a final method.

### Question 7

Which of the following statements are true?

- a. If an accessible superclass method is static, then any method with the same signature in a subclass must also be static.
- b. If a superclass method is synchronized, then the overriding method must also be synchronized.
- c. If a superclass method is public, then the overriding method must also be public.
- d. If a superclass method is native, then the overriding method must also be native.
- e. If a superclass method is protected, then the overriding method must be protected or public.

| No. | Answer      | Remark   |
|-----|-------------|--|
| 1   | e volatile  | An abstract method declaration provides no method body. If one abstract method appears within a class declaration, then the entire class must be declared abstract and the class can not be instantiated. The access modifiers, private, protected and public, can be applied to a method. The field modifiers, transient and volatile, are not applicable to method declarations. |
| 2   | d transient | A final method can not be overridden. A static method is associated with a class, but not a particular instance of the class. A thread can not enter a synchronized method without first acquiring a lock. The field modifiers, transient and volatile, are not applicable to method declarations. A native method is  |

|   |                       |  |   |
|---|-----------------------|--|---|
|   |                       |  | implemented in platform-dependent code.   |
| 3 | a<br>b<br>e<br>f<br>g | final private static synchronized native   | A final or private method can not be overridden, and can not be abstract. An abstract method declaration provides no implementation of the method, and all implementation details are left to the overriding method in the subclass. The synchronized modifier specifies an implementation detail that can be omitted from the declaration of an overriding method of a subclass, so it makes no sense to allow the use of the synchronized modifier in an abstract method declaration.           |
| 4 | e                     | None of the above.   | An abstract method of a subclass can override by an abstract method of a superclass. The overriding abstract method declaration can be a good place to add comments. An abstract method of a subclass can override a concrete implementation of a method of a superclass. An abstract method declaration can have a throws clause. The body of an abstract method is a semicolon.   |
| 5 | d                     | The keyword, super, may be used in the body of a static method.  | The keyword, this, refers to the instance on which the method has been invoked. A static method--also known as a <i>class method</i> --is not invoked on a particular instance of an object, but is instead invoked on the class. Since a static method is not associated with a particular instance, an attempt to use the keyword, this, within the body of a static method results in a Compile-time error. Similarly, the keyword, super, can not be used within the body of a static method. |
| 6 | a<br>b<br>e           | A final method can not be overridden.<br>All methods declared in a final class are implicitly final. A machine-code generator can inline the body of a final method. | All methods declared in a final class are implicitly final. It is permissible--but not required--that all such methods be explicitly declared final. All private methods are implicitly final. It is permissible--but not required--that all private methods be explicitly declared final. The body of an inline method is  |

|   |             |  |   |
|---|-------------|--|---|
|   |             | inserted directly into the code at the point where the method is invoked. If the method is invoked at 10 different points in the code, then the body can be copied to all 10 points. Inline code runs very quickly, because it removes the need to do the work associated with a method invocation. If the method is executed repeatedly in a loop, then inlining can improve performance significantly. The machine-code generator has the option to inline a final method. |   |
| 7 | a<br>c<br>e | If an accessible superclass method is static, then any method with the same signature in a subclass must also be static. If a superclass method is public, then the overriding method must also be public. If a superclass method is protected, then the overriding method must be protected or public.  | The signature of a method is the name of the method and the number and types of the method parameters. The return type is not part of the method signature. An instance method declared in a subclass overrides an accessible superclass method with the same signature. A static method of a subclass hides--but does not override--an accessible superclass method with the same signature. A compile-time error is generated if a subclass contains a declaration of an instance method that shares the same signature as an accessible static method of the superclass. The modifiers, synchronized, native and strictfp, specify implementation details that the programmer is free to change in a subclass. Similarly, a subclass can override a concrete implementation of a method with an abstract method declaration. A subclass method may not have weaker access than the overridden superclass method. For example, a public method may not be overridden by a private method. However, a subclass method can provide greater access than a superclass method. For example, a protected method can be overridden by a public method. |

| No. | Answer | Remark |
|-----|--------|--------|
|-----|--------|--------|

|   |                       |   |   |
|---|-----------------------|---|---|
|   |                       |   | An abstract method declaration provides no method body. If one abstract method is appears within a class declaration, then the entire class must be declared abstract and the class can not be instantiated. The access modifiers, private, protected and public, can be applied to a method. The field modifiers, transient and volatile, are not applicable to method declarations.   |
| 1 | e                     | volatile  | A final method can not be overridden. A static method is associated with a class, but not a particular instance of the class. A thread can not enter a synchronized method without first acquiring a lock. The field modifiers, transient and volatile, are not applicable to method declarations. A native method is implemented in platform-dependent code.   |
| 2 | d                     | transient   | A final or private method can not be overridden, and can not be abstract. An abstract method declaration provides no implementation of the method, and all implementation details are left to the overriding method in the subclass. The synchronized modifier specifies an implementation detail that can be omitted from the declaration of an overriding method of a subclass, so it makes no sense to allow the use of the synchronized modifier in an abstract method declaration. |
| 3 | a<br>b<br>e<br>f<br>g | final private static synchronized<br>native                     | An abstract method of a subclass can override by an abstract method of a superclass. The overriding abstract method declaration can be a good place to add comments. An abstract method of a subclass can override a concrete implementation of a method of a superclass. An abstract method declaration can have a throws clause. The body of an abstract method is a semicolon.   |
| 4 | e                     | None of the above.  | The keyword, this, refers to the instance on which the method has been invoked. A static method--also known as a <i>class</i>   |
| 5 | d                     | The keyword, super, may be used in the body of a static method. |   |

|   |  |   |
|---|--|---|
|   |  | <p><i>method--</i> is not invoked on a particular instance of an object, but is instead invoked on the class. Since a static method is not associated with a particular instance, an attempt to use the keyword, <i>this</i>, within the body of a static method results in a Compile-time error. Similarly, the keyword, <i>super</i>, can not be used within the body of a static method.</p>   |
| 6 | <p>A final method can not be overridden.</p> <p>a All methods declared in a final class are implicitly final. A machine-code generator can inline the body of a final method.</p>  | All methods declared in a final class are implicitly final. It is permissible--but not required--that all such methods be explicitly declared final. All private methods are implicitly final. It is permissible--but not required--that all private methods be explicitly declared final. The body of an inline method is inserted directly into the code at the point where the method is invoked. If the method is invoked at 10 different points in the code, then the body can be copied to all 10 points. Inline code runs very quickly, because it removes the need to do the work associated with a method invocation. If the method is executed repeatedly in a loop, then inlining can improve performance significantly. The machine-code generator has the option to inline a final method. |
| 7 | <p>If an accessible superclass method is static, then any method with the same signature in a subclass must also be static. If a superclass method is public, then the overriding method must also be public. If a superclass method is protected, then the overriding method must be protected or public.</p> | The signature of a method is the name of the method and the number and types of the method parameters. The return type is not part of the method signature. An instance method declared in a subclass overrides an accessible superclass method with the same signature. A static method of a subclass hides--but does not override--an accessible superclass method with the same signature. A compile-time error is generated if a subclass contains a declaration of an instance method that shares the same signature as an accessible static method of the superclass. The modifiers, synchronized, native and strictfp, specify implementation details that the   |

programmer is free to change in a subclass. Similarly, a subclass can override a concrete implementation of a method with an abstract method declaration. A subclass method may not have weaker access than the overridden superclass method. For example, a public method may not be overridden by a private method. However, a subclass method can provide greater access than a superclass method. For example, a protected method can be overridden by a public method.

## MOCKEXAM9 RETURN TYPES

### Question 1

```
class JSC201 {  
    static byte m1() {  
        final char c1 = '\u0001';  
        return c1; // 1  
    }  
    static byte m2(final char c2) {return c2;} // 2  
    public static void main(String[] args) {  
        char c3 = '\u0003';  
        System.out.print(""+m1()+m2(c3)); // 3  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 13
- b. Prints: 4
- c. Compile-time error at 1
- d. Compile-time error at 2
- e. Run-time error
- f. None of the above

### Question 2

```
class JSC202 {  
    static byte m1() {final short s1 = 2; return s1;} // 1
```

```
static byte m2(final short s2) {return s2;}      // 2
public static void main(String[] args) {
    short s3 = 4;
    System.out.print(""+m1()+m2(s3));           // 3
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 24
- b. Prints: 6
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Run-time error
- f. None of the above

### Question 3

```
class JSC203 {
    static int m1(byte b) {return b;} // 1
    static int m2(char c) {return c;} // 2
    static int m3(long l) {return l;} // 3
    public static void main(String[] args) {
        byte b = 1; char c = '\u0002'; long l = 4L;
        System.out.print(""+m1(b)+m2(c)+m3(l));
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 124
- b. Prints: 7
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Run-time error

### Question 4

```
class JSC204 {
    static int m1(short s) {return s;} // 1
    static int m2(float f) {return f;} // 2
    public static void main(String[] args) {
        short s = 3; float f = 5.0f;
        System.out.print(""+m1(s)+m2(f));
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 35.0

- b. Prints: 8.0
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Run-time error
- f. None of the above

### Question 5

```
class JSC205 {
    static int m1(int i) {return i;} // 1
    static void m2(int i) {return i;} // 2
    static int m3(int i) {return; } // 3
    public static void main(String[] args) {
        System.out.print(""+m1(1)+m2(2)+m3(3)); // 4
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 123
- b. Prints: 6
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.

| No. | Answer                       | Remark   |
|-----|------------------------------|--|
| 1   | d<br>Compile-time error at 2 | <p>There is a compile-time error at 2. The char type variable c2 is not a compile-time constant, so it can not be assigned to type byte without an explicit cast. The method parameter c2 is declared final, so the value of c2 can not be changed within method m2. The value of method parameter c2 is set at run time to the value of the argument that is provided when m2 is invoked at line 3. For that reason, the method parameter c2 is not a compile-time constant. In method m2, the statement, "return c2;", is a return statement with an expression, c2. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of method m2 is byte. The return statement attempts to return the value of the char type variable c2. If a char value is a compile-time constant, and if the value falls within the range of type byte, then the char value is assignable to type byte. In method m2, variable c2 is not a compile-time constant, because the value of c2 is not known at compile time. Instead, the value of c2 is assigned at run time to the value of the argument. Since the char type variable c2 is not a compile-time constant, the value of variable c2 is not assignable to the return type of method m2</p> |

|   |   |  |
|---|---|--|
|   |   | without an explicit cast. While the declaration of method m2 produces a compile-time error, the declaration of method m1 does not. The local variable c1 is declared final and the value is set at compile time; so c1 is a compile-time constant. The value \u0001 falls within the range of type byte; so the value of the compile-time constant c1 is assignable to the return type of method m1 without an explicit cast.  |
| 2 | d | Compile-time error at 2.<br><br>There is a compile-time error at 2. The short type variable s2 is not a compile-time constant, so it can not be assigned to type byte without an explicit cast. The method parameter s2 is declared final, so the value of s2 can not be changed within method m2. The value of method parameter s2 is set at run time to the value of the argument that is provided when m2 is invoked at line 3. For that reason, the method parameter s2 is not a compile-time constant. In method m2, the statement, "return s2;", is a return statement with an expression, s2. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of method m2 is byte. The return statement attempts to return the value of the short type variable s2. If a short value is a compile-time constant, and if the value falls within the range of type byte, then the short value is assignable to type byte without an explicit cast. In method m2, variable s2 is not a compile-time constant, because the value of s2 is not known at compile time. Instead, the value of s2 is assigned at run time to the value of the argument. Since the short type variable s2 is not a compile-time constant, the value of variable s2 is not assignable to the return type of method m2 without an explicit cast. While the declaration of method m2 produces a compile-time error, the declaration of method m1 does not. The local variable s1 is declared final and the value is set at compile time; so s1 is a compile-time constant. The value 2 falls within the range of type byte; so the value of the compile-time constant s1 is assignable to the return type of method m1 without an explicit cast. |
| 3 | e | Compile-time error at 3.<br><br>There is a compile-time error at line 3. The long type variable, l, can not be assigned to type int without an explicit cast. The statement, "return l;", is a return statement with an expression, l. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of the method, m3, is int. The type of the variable, l, is long, so an explicit cast is needed to perform the narrowing primitive conversion, "return (int)l;". The declarations of methods m1 and m2 do not generate compile-time errors, because the types of the   |

|   |             |   |   |
|---|-------------|---|---|
|   |             |   | expressions contained in the return statements are assignable to type int. Widening conversions from types byte, char, or short to type int do not require an explicit cast.  |
| 4 | d           | Compile-time error at 2.  | There is a compile-time error at 2, because a narrowing primitive conversion from type float to type int requires an explicit cast. There is no compile-time error at 1, because widening primitive conversions from types byte, char, or short to type int do not require an explicit cast.  |
| 5 | d<br>e<br>f | Compile-time error at 2. Compile-time error at 3.<br>Compile-time error at 4. | At line 2, the statement, "return i;", contains the expression, i. The enclosing method, m2, is declared void. The return statement generates a compile-time error, because it contains an expression. At line 3, the statement, "return;", does not contain an expression. The enclosing method, m3, is declared with the result type, int. The return statement generates a compile-time error, because it does not contain an expression that produces a value that is assignable to the declared result type. |

## MOCKEXAM SECTION 2

### FLOWCONTROL,ASSERTIONS,EXCEPTION HANDLING

#### Question 1

```
class JMM102 {
    public static void main(String args[]) {
        for (int i = 0; i<5 ;i++) {
            switch(i) {
                case 0: System.out.print("v ");break;
                case 1: System.out.print("w ");
                case 2: System.out.print("x ");break;
                case 3: System.out.print("y ");
                case 4: System.out.print("z ");break;
                default: System.out.print("d ");
            }
        }
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: v w x y z
- b. Prints: v w x y z d
- c. Prints: v w x x y z z
- d. Prints: v w w x y y z d
- e. Prints: d d d d d d
- f. Run-time error
- g. Compile-time error
- h. None of the above

## Question 2

```
class JMM103 {  
    public static void main(String args[]) {  
        for (int i = 0; i < 5 ;i++) {  
            switch(i) {  
                0: System.out.print("v ");break;  
                1: System.out.print("w ");  
                2: System.out.print("x ");break;  
                3: System.out.print("y ");  
                4: System.out.print("z ");break;  
            }}}}}
```

What is the result of attempting to compile and run the program?

- a. Prints: v w x y z
- b. Prints: v w x x y z z
- c. Prints: v w w x y y z
- d. Run-time error.
- e. Compile-time error.
- f. None of the above.

## Question 3

```
class JMM107 {  
    public static void main(String[] args) {  
        boolean b = true;  
        if (b = false) {System.out.print("A");}  
        } else if (b) {System.out.print("B");}  
        } else {System.out.print("C");}  
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Run-time error
- e. Compile-time error
- f. None of the above

## Question 4

```
class JMM108 {  
    static boolean b;
```

```
public static void main(String[] args) {  
    if (b) {System.out.print("A");}  
    } else if (b = false) {System.out.print("B");}  
    } else if (b) {System.out.print("C");}  
    } else if (!b) {System.out.print("D");}  
    } else {System.out.print("E");}  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Prints: D
- e. Prints: E
- f. Run-time error
- g. Compile-time error
- h. None of the above

### Question 5

```
class JMM109 {  
    public static void main(String[] args) {  
        boolean b;  
        if (b = false) {System.out.print("A");}  
        } else if (b) {System.out.print("B");}  
        } else if (!b) {System.out.print("C");}  
        } else {System.out.print("D");}  
    }
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Prints: D
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 6

```
class JMM122 {  
    public static void main (String[] args) {  
        for (int i = 0; i < 4; i++) {  
            switch (i) {  
                case 0: System.out.print("A");  
                case 1: System.out.print("B");  
                case 2: System.out.print("C");  
                case 3: System.out.print("D");  
            }  
        }  
    }
```

```
    case 1: System.out.print("B");
    case 2: System.out.print("C");
}}}}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABC
- b. Prints: ABCC
- c. Prints: CBA
- d. Prints: ABCBCC
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 7

```
class JMM123 {
    public static void main (String args[]) {
        int i = 0, j = 0, k = 0;
label1:
    for (int h = 0; h < 6; h++) {
label2:
    do { i++; k = h + i + j;
        switch (k) {
            default: break label1;
            case 1: continue label2;
            case 2: break;
            case 3: break label2;
            case 4: continue label2;
            case 5: continue label1;
        }
    } while (++j<5);
}
System.out.println(h + "," + i + "," + j);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0
- b. Prints: 0,2,1
- c. Prints: 1,3,1
- d. Prints: 2,4,1
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 8

```
class JMM103 {  
    public static void main(String args[]) {  
        byte b = -1;  
        switch(b) {  
            case 0: System.out.print("zero "); break;  
            case 100: System.out.print("100 "); break;  
            case 1000: System.out.print("1000 "); break;  
            default: System.out.print("Default ");  
        }  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: zero
- b. Prints: 100
- c. Prints: 1000
- d. Prints: Default
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 9

```
class JMM104 {  
    public static void main (String args[]) {  
        char c = 'b';  
        switch(c) {  
            case 'a': System.out.print("1");  
            case 'b': System.out.print("2");  
            case 'c': System.out.print("3");  
            default: System.out.print("4");  
        }  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3
- b. Prints: 34
- c. Prints: 234
- d. Prints: 1234
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 10

```
class JMM105 {  
    public static void main(String args[]) {  
        int x = 6; int success = 0;
```

```

do {
    switch(x) {
        case 0: System.out.print("0"); x += 5; break;
        case 1: System.out.print("1"); x += 3; break;
        case 2: System.out.print("2"); x += 1; break;
        case 3: System.out.print("3"); success++; break;
        case 4: System.out.print("4"); x -= 1; break;
        case 5: System.out.print("5"); x -= 4; break;
        case 6: System.out.print("6"); x -= 5; break;
    }
} while ((x != 3) || (success < 2));
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 60514233
- b. Prints: 6152433
- c. Prints: 61433
- d. Prints: 6143
- e. Run-time error
- f. Compile-time error

### Question 11

```

class JMM106 {
    public static void main(String args[]) {
        int x = -5; int success = 0;
        do {
            switch(x) {
                case 0: System.out.print("0"); x += 5; break;
                case 1: System.out.print("1"); x += 3; break;
                case 2: System.out.print("2"); x += 1; break;
                case 3: System.out.print("3"); success++; break;
                case 4: System.out.print("4"); x -= 1; break;
                case 5: System.out.print("5"); x -= 4; break;
                case 6: System.out.print("6"); x -= 5; break;
                default: x += x < 0 ? 2 : -2;
            }
        } while ((x != 3) || (success < 2));
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 60514233
- b. Prints: 1433
- c. Prints: 61433
- d. Prints: 051433
- e. Run-time error

- f. Compile-time error

### Question 12

```
class JMM124 {  
    public static void main(String args[]) {  
        int k;  
        for (int i=0, j=0; i<2; i++,j++) {System.out.print(i);} // 1  
        for (int i=0, k=0; i<2; i++,k++) {System.out.print(i);} // 2  
        for (int i=0, int j=0; i<2; i++,j++) {System.out.print(i);} // 3  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 012345
- b. Prints: 010101
- c. Compile-time error at line 1
- d. Compile-time error at line 2
- e. Compile-time error at line 3
- f. Run-time error

### Question 13

```
class JMM125 {  
    static int i;  
    public static void main(String args[]) {  
        for (i=1; i<3; i++) {System.out.print(i);} // 1  
        for (int i=1; i<3; i++) {System.out.print(i);} // 2  
        int i; // 3  
        for (i=0; i<2; i++) {System.out.print(i);} // 4  
        System.out.print(JMM125.i);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 1212010
- b. Prints: 1212013
- c. Compile-time error at line 1
- d. Compile-time error at line 2
- e. Compile-time error at line 4
- f. Run-time error
- g. None of the above

### Question 14

```
class JMM126 {
```

```
static int i;
public static void main(String args[]) {
    for (i=1; i<3; i++) {System.out.print(i);} // 1
    for (int i=1; i<3; i++) {System.out.print(i);} // 2
    int i;                                // 3
    for (int i=0; i<2; i++) {System.out.print(i);} // 4
    System.out.print(JMM126.i);
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1212010
- b. Prints: 1212013
- c. Compile-time error at line 1
- d. Compile-time error at line 2
- e. Compile-time error at line 4
- f. Run-time error
- g. None of the above

### Question 15

```
class JMM101 {
    public static void main(String[] args) {
        int i = 0;
        while (i++ < args.length) {
            System.out.print(args[i]);
        }
    }
}
```

java JMM101 A B C D E F

What is the result of attempting to compile and run the program using the specified command line?

- a. Prints: JMM101ABCDEF
- b. Prints: ABCDEF
- c. Compile-time error
- d. Run-time error
- e. None of the above

### Question 16

```
class JMM110 {
    public static void main (String[] args) {
        int j = 0;
        do for (int i = 0; i++ < 2;) {
            System.out.print(i);
        }
        while (j++ < 2);
    }
}
```

```
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 001122
- f. Prints: 1112
- g. Prints: 111222
- h. Prints: 121212
- i. Run-time error
- j. Compile-time error
- k. None of the above

### Question 17

```
class JMM111 {  
    public static void main (String[] args) {  
        int j = 0;  
        for (int i = 0; i < 2; i++) do  
            System.out.print(i);  
        while (j++ < 2);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 001122
- f. Prints: 1112
- g. Prints: 111222
- h. Prints: 121212
- i. Run-time error
- j. Compile-time error
- k. None of the above

### Question 18

```
class JMM112 {  
    public static void main (String[] args) {  
        int j = 0;  
        for (int i = 0; i++ < 2;) do  
            System.out.print(i);  
    }  
}
```

```
        while (j++ < 2);
    }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 001122
- f. Prints: 1112
- g. Prints: 111222
- h. Prints: 121212
- i. Run-time error
- j. Compile-time error
- k. None of the above

### Question 19

```
class JMM113 {
    public static void main (String[] args) {
        int i = 0, j = 0, k = 0;
        do while (i++ < 3)
            System.out.print(k++);
        while (j++ < 3);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 001122
- f. Prints: 1112
- g. Prints: 111222
- h. Prints: 121212
- i. Run-time error
- j. Compile-time error
- k. None of the above

### Question 20

```
class JMM114 {
    public static void main (String[] args) {
        int i = 0, j = 0;
        while (i++ < 3) do
```

```
        System.out.print(j);
        while (j++ < 3);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 001122
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 1112
- f. Prints: 111222
- g. Prints: 121212
- h. Run-time error
- i. Compile-time error
- j. None of the above

### Question 21

```
class JMM115 {
    static int m1(String s, int i) {
        System.out.print(s + i);
        return i;
    }
    public static void main (String[] args) {
        int i = 0, j = 0, k = 0;
        do while (m1("i", ++i) < 2)
            System.out.print("k" + ++k);
        while (m1("j", ++j) < 2);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: i1k1i2k2j1i3j2
- b. Prints: i1k1i2k2j1i1k1i2k2j2
- c. Prints: i1k1i2j1i3j2
- d. Run-time error
- e. Compile-time error
- f. None of the above

### Question 22

```
class JMM116 {
    static int m1(String s, int i) {
        System.out.print(s + i);
        return i;
    }}
```

```
public static void main (String[] args) {  
    int j = 0;  
    for (int i = m1("A",0); m1("B",i) < 2; m1("C",++i)) {  
        m1("J",++j);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A0B0C1J1B1C2J2B2
- b. Prints: A0B0J1C1B1J2C2B2
- c. Prints: A0B0J1C1A1B1J2C2A2B2
- d. Run-time error
- e. Compile-time error
- f. None of the above

### Question 23

```
class JMM117 {  
    public static void main (String[] args) {  
        int i = 0, j = 9;  
        do {  
            i++;  
            if (j-- < i++) {break;}  
        } while (i < 5);  
        System.out.print(i + "," + j);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 5,4
- b. Prints: 6,3
- c. Prints: 6,6
- d. Prints: 7,2
- e. Run-time error
- f. Compile-time error
- g. None of the above

### Question 24

```
class JMM118 {  
    public static void main (String[] args) {  
        int i = 0, j = 9;  
        while (i++ <= j--) {i++; if (j < 5) break;}  
        System.out.print(i + "," + j);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 4,7
- b. Prints: 6,6
- c. Prints: 7,2
- d. Prints: 8,5
- e. Prints: 9,4
- f. Run-time error
- g. Compile-time error
- h. None of the above

### Question 25

```
class JMM119 {  
    public static void main (String[] args) {  
        int i = 0, j = 9;  
        do {  
            if (j < 4) {break;} else if (j-- < 7) {continue;}  
            i++;  
        } while (i++ < 7);  
        System.out.print(i + "," + j);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 4,7
- b. Prints: 6,6
- c. Prints: 6,5
- d. Prints: 6,4
- e. Prints: 7,5
- f. Prints: 8,4
- g. Run-time error
- h. Compile-time error
- i. None of the above

### Question 26

```
class JMM120 {  
    public static void main (String args[]) {  
        int i = 0, j = 0, k = 0;  
        label1:  
        for (;;) { i++;  
        label2:  
        do {  
            k = i + j;  
            switch (k) {  
                case 0: continue label2;  
                case 1: continue label1;
```

```

        case 2: break;
        case 3: break label2;
        case 4: break label1;
        default: break label1;
    }
} while (++j<5);
}
System.out.println(i + "," + j);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 2,1
- b. Prints: 2,2
- c. Prints: 3,1
- d. Prints: 3,2
- e. Prints: 3,3
- f. Run-time error
- g. Compile-time error
- h. None of the above

### Question 27

```

class JMM121 {
    public static void main (String args[]) {
        int h = 0, i = 0, j = 0, k = 0;
label1:
    for (;;) { h++;
label2:
    do { i++; k = h + i + j;
        switch (k) {
            default: break label1;
            case 1: continue label1;
            case 2: break;
            case 3: break label2;
            case 4: continue label2;
            case 5: continue label1;
        }
    } while (++j < 5);
}
System.out.println(h + "," + i + "," + j);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,2,3
- b. Prints: 1,3,2
- c. Prints: 2,2,2

- d. Prints: 2,4,1
- e. Prints: 2,4,2
- f. Run-time error
- g. Compile-time error
- h. None of the above

| No. | Answer                  | Remark  |
|-----|-------------------------|---|
| 1   | c Prints: v w x x y z z | Cases one and three have no break statement, so the next case is also executed and x and z are printed twice.   |
| 2   | e Compile-time error.   | The keyword, case, is missing from the case labels.   |
| 3   | c Prints: C             | The boolean type variable, b, is initialized to true. The boolean expression of the first if statement, <code>b = false</code> , is a simple assignment expression that sets b to false. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator ( <code>==</code> ) has not been replaced by the simple assignment operator ( <code>=</code> ).   |
| 4   | d Prints: D             | The expression, <code>b = false</code> , appears to be testing the value of b, but it is really setting the value of b. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator ( <code>==</code> ) has not been replaced by the simple assignment operator ( <code>=</code> ).  |
| 5   | c Prints: C             | The first if statement initializes the value of b. The expression, <code>b = false</code> , appears to be testing the value of b, but it is really setting the value of b. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator ( <code>==</code> ) has not been replaced by the simple assignment operator ( <code>=</code> ).   |
| 6   | d Prints: ABCBCC        | The switch statement does not contain any break statements. The first case falls through to the second. The second case falls through to the third. There is no default switch label, so nothing is printed when variable i is 3.   |
| 7   | f Compile-time error    | The loop variable, h, is local to the for statement. The print statement causes a compile-time error, because it refers to the out-of-scope local variable, h. Always check for variables that are out-of-scope!  |
| 8   | f Compile-time error    | The case constant expression, 1000, produces a compile-time error, because it exceeds the range of the switch expression type, byte. The type of a switch expression can be byte, short, int or char. A constant expression is associated with each case label. Each case constant expression must be assignable to the type of the switch expression. In this case, the switch expression, b, is of type byte; so the maximum positive value of each case constant expression is limited to the maximum byte value, 127. |
| 9   | c Prints: 234           | No break statements appear inside the switch statement, so more than one case is processed.   |
| 10  | c Prints: 61433         | On the first pass through the loop, the value of x is 6, so 5 is subtracted from x. On the second pass, the value of x is 1, so 3 is  |

|    |   |  |  |
|----|---|--|--|
|    |   |  | added to x. On the third pass, the value of x is 4, so 1 is subtracted from x. On the fourth pass, the value of x is 3, so the variable, success, is incremented from zero to one. On the final pass, the value of x is 3 and the variable, success, is incremented to the value, 2. The boolean expression of the do loop is now false, so control passes out of the loop.  |
| 11 | b Prints: 1433  |  | On the first pass through the loop, the switch expression, x, has the value, -5. None of the case constants are matched, so the statement following the default label is executed causing the value of x to be set to -3. On the second pass, the default case of the switch statement is executed again, and two is again added to the value of x. The new value is -1. On the third pass, the value of x is again incremented, and the new value is +1. After that, the value of x is printed on each pass through the loop. For the last two passes, the value of x is 3.   |
| 12 | d<br>e Compile-time error at line 2<br>Compile-time error at line 3 |  | A compile-time error occurs at line 2 as a result of the attempt to shadow the method local variable, k, within the for-loop. A variable declared within the for-loop can not shadow a local variable of the enclosing method. At line 3, the declaration of variable j causes a compile-time error. The initialization part of a for statement may use a single local variable declaration statement to declare more than one local variable. Each of the new variables is declared in a comma separated list of variable declarators. In this program, the local variables should have been declared as follows: "int i=0, j=0;". Instead, the type of variable j has been incorrectly added to the declarator: "int i=0, int j=0;". The result is a compile-time error.           |
| 13 | b Prints: 1212013   |  | A method local variable, i, is declared at line 3. At line 4, the initialization part of the for statement initializes the method local variable to the value zero. The for statement does not declare a new variable, i, so the method local variable is not shadowed within the loop. Suppose a local variable, v, is declared within a statement or block. Variable, v, can shadow a class variable or an instance variable of the same name, but a compile-time error is generated if v shadows a method local variable. Please note that a compile-time error is not generated if a local variable is shadowed within the declaration of a class that is nested within the scope of the local variable.   |
| 14 | e Compile-time error at line 4                                      |  | A method local variable, i, is declared at line 3. At line 4, the initialization part of the for statement attempts to shadow the method local variable with a new variable that is intended to be local to the for statement. The result is a compile-time error. At line 2, a new variable, i, is declared within the for statement. That variable shadows the class variable, but it does not shadow the method local variable declared at line 3. The scope of a method local variable begins with its own initializer--if present--and extends to the end of the method body. At line 2, the method local variable is not in scope, so shadowing does not occur. Suppose a local variable, v, is declared within a statement or block. Variable, v, can shadow a class variable |

|    |   |                             |  |
|----|---|-----------------------------|--|
|    |   |                             | or an instance variable of the same name, but a compile-time error is generated if v shadows a method local variable. Please note that a compile-time error is not generated if a local variable is shadowed within the declaration of a class that is nested within the scope of the local variable.  |
| 15 | d | Run-time error              | The index, i, is incremented before the array access expression is evaluated, so the first element accessed is at index one instead of zero. The arguments, BCDEF, are printed before an <code>ArrayIndexOutOfBoundsException</code> is thrown when the attempt is made to access an element beyond the end of the array.  |
| 16 | h | Prints: 121212              | This trick question has a for loop nested inside of a do loop. For each iteration, the values of i and j are as follows:<br>(1,0)(2,0)(1,1)(2,1)(1,2)(2,2).  |
| 17 | a | Prints: 0001                | This trick question has a do-loop nested inside of a for-loop. For each iteration, the values of i and j are as follows: (0,0)(0,1)(0,2)(1,3).   |
| 18 | f | Prints: 1112                | This trick question has a do-loop nested inside of a for-loop. For each iteration, the values of i and j are as follows: (1,0)(1,1)(1,2)(2,3).   |
| 19 | b | Prints: 012                 | This trick question has a while-loop nested inside of a do-loop. For each iteration, the values of i, j and k are as follows:<br>(1,0,0)(2,0,1)(3,0,2).  |
| 20 | d | Prints: 012345              | This trick question has a do-loop nested inside of a while-loop. For each iteration, the values of i and j are as follows:<br>(1,0)(1,1)(1,2)(1,3)(2,4)(3,5).  |
| 21 | c | Prints: i1k1i2j1i3j2        | A while loop is nested inside of a do loop. The while loop iterates once during the first iteration of the do loop. The body of the while loop does not execute during the final iteration of the do loop.   |
| 22 | b | Prints:<br>A0B0J1C1B1J2C2B2 | The initialization statement, labeled A, is processed first. The boolean expression, labeled B, is processed before each iteration of the loop. The body of the loop is processed next followed by the update expression, labeled C.   |
| 23 | c | Prints: 6,6                 | Suppose the print statement, <code>System.out.print(" (" + i + "," + j + ")");</code> , is inserted at the top of the loop. The output of the program would be as follows: (0,9)(2,8)(4,7)6,6. The variable, i, is incremented twice with each pass through the loop. The variable, j, is decremented once with each pass. The loop terminates when i reaches six.   |
| 24 | e | Prints: 9,4                 | Suppose the print statement, <code>System.out.print(" (" + i + "," + j + ")");</code> , is inserted at the top of the loop. The output of the program would be as follows: (1,8)(3,7)(5,6)(7,5)9,4. The variable, i, is incremented twice with each pass through the loop. The variable, j, is decremented once with each pass. The boolean expression of the while loop, <code>i++ &lt;= j--</code> , is false when i reaches 8 and j reaches 5. The value of i is subsequently incremented and j is decremented yielding 9 and 4 respectively. Those values are printed. |
| 25 | f | Prints: 8,4                 | Suppose the print statement, <code>System.out.print(" (" + i + "," + j + ")");</code> , is inserted at the top of the loop. The output of the program would be as follows: (0,9)(2,8)(4,7)(6,6)(7,5)8,4. The initial value of j is 9.  |

|    |                 |   |
|----|-----------------|---|
|    |                 | With each pass through the loop, the expression, $j-- < 7$ , decrements $j$ . The initial value of variable $i$ is 0. With each pass through the loop, the expression, $i++ < 7$ , increments variable $i$ . Inside the body of the loop, $i$ is also incremented as long as $j$ is greater than or equal to 7. When $j$ is less than seven, variable $i$ is no longer incremented inside the body of the loop, but it is still incremented by the expression, $i++ < 7$ .  |
| 26 | c Prints: 3,1   | Suppose the print statement, <code>System.out.print("("+i+","+j+","+k+")");</code> , is inserted before the switch statement. The output of the program would be as follows: (1,0,1)(2,0,2)(2,1,3)(3,1,4)3,1. On the first iteration, case 1 is processed. The "continue label1;" statement causes control to go directly to the top of the for-loop following the label, label1. The boolean expression of the do-loop is not processed. On the second iteration, case 2 is processed. The break statement causes the switch statement to complete, and control passes to the boolean expression of the do-loop. On the third iteration, case 3 is processed. The break with label statement, "break label2;", completes abruptly; and the do-loop completes abruptly without processing the loop expression, $++j < 5$ , so $j$ is not incremented. On the fourth iteration, case 4 is processed. The break with label statement, "break label1;", completes abruptly, and the for-loop completes abruptly. |
| 27 | b Prints: 1,3,2 | Suppose the print statement, <code>System.out.print("("+h+","+i+","+j+","+k+")");</code> , is inserted before the switch statement. The output of the program would be as follows: (1,1,0,2)(1,2,1,4)(1,3,2,6)1,3,2. Three iterations of the loop are executed. On the first iteration, the switch expression, $k$ , matches the case constant, 2. The subsequent break statement is executed and the switch statement completes normally. On the second iteration, the switch expression matches the case constant, 4; and the subsequent continue statement causes control to pass to the loop-continuation point of the do statement where the expression, $++j < 5$ , is evaluated and found to be true. On the final iteration, the switch expression does not match any case constant; so the break statement following the default label is processed  |

## Question 1

```
class A {
    public static void main (String[] args) {
        Error error = new Error();
        Exception exception = new Exception();
        System.out.print((exception instanceof Throwable) + ",");
        System.out.print(error instanceof Throwable);
```

}

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 2

```
class A {A() throws Exception {}} // 1
class B extends A {B() throws Exception {}} // 2
class C extends A {C() {}} // 3
```

Which of the following statements are true?

- a. class A extends Object.
- b. Compile-time error at 1.
- c. Compile-time error at 2.
- d. Compile-time error at 3.

### Question 3

```
class A {
    public static void main (String[] args) {
        Object error = new Error();
        Object runtimeException = new RuntimeException();
        System.out.print((error instanceof Exception) + ",");
        System.out.print(runtimeException instanceof Exception);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 4

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 1;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                } a++;
            }
            catch (Level2Exception e) {b++;}
            finally {c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,0,0
- b. Prints: 0,0,1,1,0,1
- c. Prints: 0,1,1,1,0,1
- d. Prints: 1,0,1,1,0,1
- e. Prints: 1,1,1,1,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 5

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 2;
        try {
            try {

```

```

switch (x) {
    case 1: throw new Level1Exception();
    case 2: throw new Level2Exception();
    case 3: throw new Level3Exception();
} a++;
}
catch (Level2Exception e) {b++;}
finally {c++;}
}
catch (Level1Exception e) { d++;}
catch (Exception e) {f++;}
finally {g++;}
System.out.print(a+","+b+","+c+","+d+","+f+","+g);
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,1,0,0,1
- b. Prints: 0,1,0,0,0,0
- c. Prints: 0,1,1,0,0,1
- d. Prints: 0,1,0,0,0,1
- e. Prints: 1,1,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 6

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 3;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                } a++;
            }
            catch (Level2Exception e) {b++;}
            finally {c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
    }
}

```

```

        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 7

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 4;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                    case 4: throw new Exception();
                } a++;
            } catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,0,0,1
- b. Prints: 0,0,0,0,1,0
- c. Prints: 0,0,1,0,0,1
- d. Prints: 0,0,1,0,1,1

- e. Prints: 0,1,1,1,1,1
- f. Prints: 1,1,1,1,1,1
- g. Compile-time error
- h. Run-time error
- i. None of the above

### Question 8

```
class Level1Exception extends Exception {}  
class Level2Exception extends Level1Exception {}  
class Level3Exception extends Level2Exception {}  
class Purple {  
    public static void main(String args[]) {  
        int a,b,c,d,f,g,x;  
        a = b = c = d = f = g = 0;  
        x = 5;  
        try {  
            try {  
                switch (x) {  
                    case 1: throw new Level1Exception();  
                    case 2: throw new Level2Exception();  
                    case 3: throw new Level3Exception();  
                    case 4: throw new Exception();  
                } a++; }  
                catch (Level2Exception e) {b++;}  
                finally {c++;}  
            }  
            catch (Level1Exception e) { d++;}  
            catch (Exception e) {f++;}  
            finally {g++;}  
            System.out.print(a+" "+b+" "+c+" "+d+" "+f+" "+g);  
        }  
    }
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,0,0,0,0,0
- b. Prints: 1,0,1,0,0,1
- c. Prints: 0,0,1,0,0,1
- d. Prints: 1,1,1,1,1,1
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 9

```
class ColorException extends Exception {}
```

```
class WhiteException extends ColorException {}  
class White {  
    void m1() throws ColorException {throw new WhiteException();}  
    void m2() throws WhiteException {}  
    public static void main (String[] args) {  
        White white = new White();  
        int a,b,d,f; a = b = d = f = 0;  
        try {white.m1(); a++;} catch (ColorException e) {b++;}  
        try {white.m2(); d++;} catch (WhiteException e) {f++;}  
        System.out.print(a+","+b+","+d+","+f);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,0
- c. Prints: 0,1,1,0
- d. Prints: 1,1,1,0
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

#### Question 10

```
class ColorException extends Exception {}  
class WhiteException extends ColorException {}  
class White {  
    void m1() throws ColorException {throw new ColorException();}  
    void m2() throws WhiteException {throw new ColorException();}  
    public static void main (String[] args) {  
        White white = new White();  
        int a,b,d,f; a = b = d = f = 0;  
        try {white.m1(); a++;} catch (ColorException e) {b++;}  
        try {white.m2(); d++;} catch (WhiteException e) {f++;}  
        System.out.print(a+","+b+","+d+","+f);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 11

```
class ColorException extends Exception {}  
class WhiteException extends ColorException {}  
class White {  
    void m1() throws ColorException { throw new ColorException();}  
    void m2() throws WhiteException { throw new WhiteException();}  
    public static void main (String[] args) {  
        White white = new White();  
        int a,b,d,f; a = b = d = f = 0;  
        try {white.m1(); a++;} catch (WhiteException e) {b++;}  
        try {white.m2(); d++;} catch (WhiteException e) {f++;}  
        System.out.print(a+","+b+","+d+","+f);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 12

```
class Level1Exception extends Exception {}  
class Level2Exception extends Level1Exception {}  
class Level3Exception extends Level2Exception {}  
class Brown {  
    public static void main(String args[]) {  
        int a, b, c, d, f; a = b = c = d = f = 0;  
        int x = 1;  
        try {  
            switch (x) {  
                case 1: throw new Level1Exception();  
                case 2: throw new Level2Exception();  
                case 3: throw new Level3Exception();  
            } a++; }  
        catch (Level3Exception e) {b++;}  
        catch (Level2Exception e) {c++;}  
        catch (Level1Exception e) {d++;}  
        finally {f++;}
```

```
        System.out.print(a+"."+b+"."+c+"."+d+"."+f);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

### Question 13

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
        int a, b, c, d, f; a = b = c = d = f = 0;
        int x = 2;
        try {
            switch (x) {
                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            } a++;
        } catch (Level3Exception e) {b++;}
        catch (Level2Exception e) {c++;}
        catch (Level1Exception e) {d++;}
        finally {f++;}
        System.out.print(a+"."+b+"."+c+"."+d+"."+f);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error

- i. Run-time error
- j. None of the above

#### Question 14

```
class Level1Exception extends Exception {}  
class Level2Exception extends Level1Exception {}  
class Level3Exception extends Level2Exception {}  
class Brown {  
    public static void main(String args[]) {  
        int a, b, c, d, f; a = b = c = d = f = 0;  
        int x = 4;  
        try {  
            switch (x) {  
                case 1: throw new Level1Exception();  
                case 2: throw new Level2Exception();  
                case 3: throw new Level3Exception();  
            } a++; }  
        catch (Level3Exception e) {b++;}  
        catch (Level2Exception e) {c++;}  
        catch (Level1Exception e) {d++;}  
        finally {f++;}  
        System.out.print(a+","+b+","+c+","+d+","+f);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

#### Question 15

```
class ColorException extends Exception {}  
class WhiteException extends ColorException {}  
abstract class Color {  
    abstract void m1() throws ColorException;  
}  
class White extends Color {
```

```

void m1() throws WhiteException {throw new WhiteException();}
public static void main (String[] args) {
    White white = new White();
    int a,b,c; a = b = c = 0;
    try {white.m1(); a++;}
    catch (WhiteException e) {b++;}
    finally {c++;}
    System.out.print(a+","+b+","+c);
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,1
- c. Prints: 0,1,0
- d. Prints: 0,1,1
- e. Prints: 1,0,0
- f. Prints: 1,0,1
- g. Prints: 1,1,0
- h. Prints: 1,1,1
- i. Compile-time error
- j. Run-time error
- k. None of the above

### Question 16

```

class RedException extends Exception {}
class BlueException extends Exception {}
class White {
    void m1() throws RedException {throw new RedException();}
    public static void main (String[] args) {
        White white = new White();
        int a,b,c,d; a = b = c = d = 0;
        try {white.m1(); a++;}
        catch (RedException e) {b++;}
        catch (BlueException e) {c++;}
        finally {d++;}
        System.out.print(a+","+b+","+c+","+d);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error

- g. Run-time error
- h. None of the above

### Question 17

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 1;
        try {
            throw new Level1Exception();
        try {
            switch (x) {
                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            } a++;
        } catch (Level2Exception e) {b++;}
        finally {c++;}
    }
    catch (Level1Exception e) { d++;}
    catch (Exception e) {f++;}
    finally {g++;}
    System.out.print(a+"," +b+"," +c+"," +d+"," +f+"," +g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

| No. | Answer | Remark   |
|-----|--------|--|
| 1   | d      | Prints: true,true<br>Both Error and Exception are subclasses of Throwable.   |
| 2   | a<br>d | class A extends<br>Object.<br>Compile-time<br>The constructors for class B and class C both invoke the<br>constructor for A. The constructor for class A declares Exception<br>in the throws clause. Since the constructors for B and C invoke |

|    |   |                        |   |
|----|---|------------------------|---|
|    |   | error at 3.            | the constructor for A, it is necessary to declare Exception in the throws clauses of B and C. A compile-time error is generated at marker 3, because the constructor does not declare Exception in the throws clause.   |
| 3  | b | Prints: false,true     | Error is a direct subclass of Throwable. RuntimeException is a direct subclass of Exception.  |
| 4  | b | Prints:<br>0,0,1,1,0,1 | The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws a Level1Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the first of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.   |
| 5  | c | Prints:<br>0,1,1,0,0,1 | The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.  |
| 6  | b | Prints:<br>0,1,1,0,0,1 | The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.  |
| 7  | d | Prints:<br>0,0,1,0,1,1 | The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws an Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the second of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.   |
| 8  | b | Prints:<br>1,0,1,0,0,1 | The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. Both of the finally blocks are then executed.  |
| 9  | c | Prints: 0,1,1,0        | The first try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. The throws clause of m1 declares a ColorException, so the body may throw a ColorException or any subclass of ColorException. The second try block also contains two statements. The first invokes method m2, and the subsequent statement contains a post increment expression with the variable, d, as the operand. Method m2 does not throw an exception, so d is incremented, and the try block completes normally. Although the throws clause of m2 declares a WhiteException, there is no requirement to throw any exception. |
| 10 | f | Compile-time error     | The throws clause of White.m2 declares a WhiteException, so the body of m2 may throw a WhiteException or any subclass of WhiteException. Instead, the body of m2 throws a superclass of WhiteException. The result is a compile-time error.   |

|    |   |                    |   |
|----|---|--------------------|---|
| 11 | f | Compile-time error | The throws clause of White.m1 declares a ColorException, but the catch clause in the main method catches only a subclass of ColorException. The result is a compile-time error.   |
| 12 | a | Prints: 0,0,0,1,1  | The first catch clause has a parameter e of type Level3Exception, so the first catch clause is able to catch any exception type that is assignable to type Level3Exception. Since Level2Exception is the superclass of Level3Exception, an instance of Level2Exception is not assignable to a catch clause parameter of type Level3Exception. Similarly, Level1Exception is also a superclass of Level3Exception, so an instance of Level1Exception is not assignable to a catch clause parameter of type Level3Exception. The only exception type that can be caught by the first catch clause is a Level3Exception. The second catch clause has a parameter e of type Level2Exception, so the second catch clause is able to catch a Level2Exception. The Level1Exception is the superclass of Level2Exception. An instance of Level1Exception is not assignable to a catch clause parameter of type Level2Exception, so the second catch clause can not catch a Level1Exception. Since a Level3Exception is a subclass of Level2Exception an exception of type Level3Exception is assignable to a catch clause parameter type Level2Exception. All exceptions of type Level3Exception will be caught by the first catch clause, so the second catch clause in this program will not have an opportunity to catch a Level3Exception. The third catch clause has a parameter e of type Level1Exception, so the third catch clause is able to catch a Level1Exception. The exceptions of type Level2Exception and Level3Exception are assignable to the catch clause parameter of the third catch clause, but the exceptions of those subclass types will be caught by the first two catch clauses. The switch statement throws a Level1Exception. The try block completes abruptly as control passes to the third catch block where d is incremented. The finally block is also executed, so f is incremented. |
| 13 | e | Prints: 0,0,1,0,1  | The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception. The switch statement throws a Level2Exception. The try block completes abruptly as control passes to the second catch block where c is incremented. The finally block is also executed, so f is incremented.   |
| 14 | g | Prints: 1,0,0,0,1  | The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. The finally block is also executed, so f is incremented.   |
| 15 | d | Prints: 0,1,1      | The try block contains two statements. The first invokes method   |

|    |   |  |
|----|---|--|
|    |   | m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. Although Color.m1 declares a ColorException in the throws clause, a subclass of Color is free to declare only a subclass of ColorException in the throws clause of the overriding method. |
| 16 | f | Compile-time error<br>A compile-time error is generated, because the second catch clause attempts to catch an exception that is never thrown in the try block.   |
| 17 | f | Compile-time error<br>A throw statement is the first statement in the outer try block. A throw statement appearing in a try block causes control to pass out of the block. Consequently, statements can not be reached if they appear in the block after the throw statement. The switch statement that appears after the throw statement is unreachable and results in a compile-time error.  |

### Question 1

Which of the following are command-line switches used to enable assertions in non-system classes?

- a. -ea
- b. -ae
- c. -aon
- d. -aoff
- e. -enableassertions
- f. -assertionsenable
- g. -assertionson
- h. -assertionsoff

### Question 2

Which of the following are command-line switches used to disable assertions in non-system classes?

- a. -da
- b. -ad
- c. -aon
- d. -aoff
- e. -disableassertions
- f. -assertionsdisable

- g. -assertionson
- h. -assertionsoff

### Question 3

Which of the following are command-line switches used to disable assertions in non-system classes?

- a. -disableassertions
- b. -disableAssertions
- c. -assertionsDisable
- d. -assertionsOn
- e. -assertionsOff
- f. None of the above

### Question 4

Which of the following are command-line switches used to enable assertions in system classes?

- a. -saon
- b. -saoff
- c. -esa
- d. -sae
- e. -systemassertionson
- f. -systemassertionsoff
- g. -enablesystemassertions
- h. -systemassertionenable

### Question 5

Which of the following statements are true?

- a. By default assertions are disabled at run-time.
- b. Assertions can be selectively enabled for any named class.
- c. If assertions are selectively enabled for a named class then assertions are automatically enabled for any subclass of the named class.
- d. Assertions can be selectively enabled for any named package.
- e. If assertions are selectively enabled for any named package then assertions are automatically enabled for the subpackages.
- f. Assertions can be selectively enable for the unnamed package in the current working directory.
- g. Assertions can be selectively enable for any unnamed package in any named directory.

## Question 6

Which of the following is thrown to indicate that an assertion has failed?

- a. `AssertionError`
- b. `AssertException`
- c. `AssertionError`
- d. `AssertionException`
- e. None of the above

## Question 7

```
class A {  
    void m1(int i) {  
        int j = i % 3;  
        switch (j) {  
            case 0: System.out.print("0"); break;  
            case 1: System.out.print("1"); break;  
            default:  
                assert j == 2;  
                System.out.print(j);  
        }  
    }  
    public static void main (String[] args) {  
        A a = new A();  
        for (int i=5; i >= -1; i--) {a.m1(i);}  
    }  
}
```

Which statements are true?

- a. With assertions enabled it prints 210210-1 followed by an `AssertionError` message.
- b. With assertions enabled it prints 210210 followed by an `AssertionError` message.
- c. With assertions enabled it prints only 210210.
- d. With assertions enabled it prints nothing.
- e. With assertions disabled it prints 210210-1
- f. With assertions disabled it prints only 210210
- g. Assertions should not be used within the default case of a switch statement.

## Question 8

```
class B {  
    private static int x1;  
    public void setX(int x) {x1 = x;}  
    public int getX() {return x1;}  
    public static void main(String[] args) {  
        int i1 = 0, j1 = 0;  
        do {  
    }
```

```
System.out.print(j1++);
assert (i1 = j1 + x1) < 6;
} while (i1 < 3);
}}
```

Assuming that the setX method is never invoked, which statements are true?

- a. With assertions enabled it prints 012345 followed by an AssertionError message.
- b. With assertions enabled it prints 012.
- c. With assertions disabled it prints: 012345
- d. With assertions disabled it prints: 012
- e. With assertions disabled it attempts to print an infinite sequence of numbers.
- f. With assertions disabled the variable i1 is incremented with each pass through the loop.
- g. As a rule, the boolean expression of an assert statement should not be used to perform actions that are required for normal operation of the program.

### Question 9

Which statements are true?

- a. Assertions should not be used to validate arguments passed to public methods.
- b. Assertions should not be used to validate arguments passed to nonpublic methods.
- c. Assertions should not be used within the default case of a switch statement.
- d. Assertions should not be used to perform processing that is required for the normal operation of the application.

### Question 10

Which statements are true?

- a. Assertions should never be placed at any point in the code that should not be reached under normal circumstances.
- b. The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification.
- c. A catch clause should not be used to catch an AssertionError.
- d. AssertionError is a checked exception.

### Question 11

```
class C {
String m1(int i) {
switch (i) {
case 0: return "A";
case 1: return "B";
case 2: return "C";
```

```

        default: throw new AssertionError();
    }
}
public static void main(String[] args) {
    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}

```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
- b. With assertions disabled it prints ABC followed by an AssertionError message.
- c. Assertions should not be used within the default case of a switch statement.
- d. In this code example an assert statement could not be used in place of the "throw" statement.

### Question 12

```

class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";
            default:
                assert false;
        }
        return "E";
    }
}
public static void main(String[] args) {
    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}

```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
- b. With assertions enabled it prints ABCE followed by an AssertionError message.
- c. With assertions disabled it prints ABC
- d. With assertions disabled it prints ABCE
- e. Assertions should not be used within the default case of a switch statement.

### Question 13

```

class C {
    String m1(int i) {

```

```

switch (i) {
    case 0: return "A";
    case 1: return "B";
    case 2: return "C";
    default:
        assert false;
}
}

public static void main(String[] args) {
    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}

```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
- b. With assertions disabled it prints ABC followed by an AssertionError message.
- c. Assertions should not be used within the default case of a switch statement.
- d. In this code example a throw statement must be used in place of the assert statement.
- e. Compile-time error

#### Question 14

```

class D {
    private boolean b1, b2;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void m1 () {
        if (!b2 & !b1) {System.out.print("A");}
        } else if (!b2 & b1) {System.out.print("B");}
        } else if (b2 & !b1) {System.out.print("C");}
        } else {assert false;}
    }
    public static void main (String[] args) {
        D d = new D();
        d.setB1(true); d.setB2(true);
        d.m1();
    }
}

```

Which statements are true?

- a. With assertions enabled it prints an AssertionError message.
- b. With assertions enabled it prints nothing.
- c. With assertions disabled it prints an AssertionError message.
- d. With assertions disabled it prints nothing.
- e. An assertion should not be placed at any location that the programmer believes will

- never be reached under normal operating conditions.
- f. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.

### Question 15

```
class A {  
    private void m1 (int i) {  
        assert i < 10 : i; System.out.print(i);  
    }  
    public void m2 (int i) {  
        assert i < 10 : i; System.out.print(i);  
    }  
    public static void main (String[] args) {  
        A a = new A(); a.m1(11); a.m2(12);  
    } }
```

Which statements are true?

- a. If assertions are enabled at run time it prints an error message.
- b. With assertions enabled it prints nothing.
- c. With assertions disabled it prints an error message.
- d. With assertions disabled it prints 1112.
- e. With assertions disabled it prints nothing.
- f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.
- g. Method m1 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a non-public method.
- h. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method.

### Question 16

```
class B {  
    int a, b, c;  
    private void setA(int i) {a = i;}  
    private void setB(int i) {b = i;}  
    private int m1 (int i) {  
        c = a + b + i; assert c < 200 : c; return c;  
    }  
    public static void main (String[] args) {  
        B b = new B(); b.setA(50); b.setB(100); b.m1(50);  
    } }
```

Which statements are true?

- a. If assertions are not enabled at run time it prints an error message.

- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.

### Question 17

```
class C {  
    int a, b, c;  
    public void setA(int i) {a = i; assert validateC() : c;}  
    public void setB(int i) {b = i; assert validateC() : c;}  
    private boolean validateC() {  
        return c > a + 2 * b;  
    }  
    public int m1(int i) {  
        c = a + b + i;  
        assert validateC() : c;  
        return c;  
    }  
    public C(int i) {  
        c = i; assert validateC() : c;  
    }  
    public static void main(String[] args) {  
        C c = new C(251); c.setA(50); c.setB(100);  
    }  
}
```

Which statements are true?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.
- f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.

### Question 18

```
class E {  
    private boolean b1, b2, b3;  
    public void setB1(boolean b) {b1 = b;}  
    public void setB2(boolean b) {b2 = b;}  
    public void setB3(boolean b) {b3 = b;}  
    public void m1 (int i) {
```

```

b2 = i % 2 == 0;
if (!b3 & !b2 & !b1) {System.out.print("A");
} else if (!b3 & !b2 & b1) {System.out.print("B");
} else if (!b3 & b2 & !b1) {System.out.print("C");
} else { // Only b3 is true.
    assert b3 & !b2 & !b1;
}
System.out.print(b1 + "," + b2 + "," + b3);
b1 = b2 = b3 = false;
}
public static void main (String[] args) {
    E e = new E(); e.setB1(true); e.m1(2);
}
}

```

Which statements are true?

- a. With assertions enabled it prints an error message.
- b. With assertions enabled it prints: true,true,false
- c. With assertions disabled it prints an error message.
- d. With assertions disabled it prints: true,true,false
- e. With assertions disabled it prints nothing.
- f. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2 or b3, to be true.
- g. The assert statement is being used to check a precondition--something that must be true when the method begins.
- h. The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program.

### Question 19

```

class F {
    private boolean b1, b2, b3;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void setB3(boolean b) {b3 = b;}
    public String m1 (int i) {
        b2 = i % 2 == 0;
        if (!b3 & !b2 & !b1) {return "None are true.";}
        } else if (!b3 & !b2 & b1) {return "Only b1 is true.";}
        } else if (!b3 & b2 & !b1) {return "Only b2 is true.";}
        } else if (b3 & !b2 & !b1) {return "Only b3 is true.";}
        } else {throw new AssertionError();}
    }
    public static void main (String[] args) {
        F f = new F();
        f.setB1(true);
        System.out.println(f.m1(5));
    }
}

```

```

        System.out.println(f.m1(6));
    }
}

```

Which statements are true?

- a. Prints "Only b1 is true" followed by an error message.
- b. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- c. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
- d. The assert statement is being used to check a precondition--something that must be true when the method begins.
- e. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
- f. The throw statement could be replaced by an assert statement.

### Question 20

An assert statement can be used to check a control-flow invariant to verify which of the following?

- a. A particular assumption is true when the flow of control enters a method.
- b. The flow of control does not reach a particular point in the program.
- c. The normal flow of control has reached a particular point in the program.
- d. The normal flow of control has reached the end of a method.
- e. The default case of a switch statement is not reached.
- f. The else block of an if/else statement is not reached.

| No. | Answer  | Remark   |
|-----|---|--|
| 1   | a<br>e -ea -enableassertions  | Two command-line switches used to enable assertions in non-system classes are -ea and -enableassertions.   |
| 2   | a<br>e -da -disableassertions   | Two command-line switches used to disable assertions are -da and -disableassertions. Remember that all of the letters are lower case.                        |
| 3   | a -disableassertions  | Two command-line switches used to disable assertions in non-system classes are -da and -disableassertions. Remember that all of the letters are lower case.  |
| 4   | c<br>g -esa -enablesystemassertions   | Two command-line switches used to enable assertions in system classes are -esa and -enablesystemassertions. Remember that all of the letters are lower case. |
| 5   | a By default assertions are disabled<br>b at run-time. Assertions can be<br>d selectively enabled for any named |  |

|   |             |   |  |
|---|-------------|---|--|
|   | e<br>f      | class. Assertions can be selectively enabled for any named package. If assertions are selectively enabled for any named package then assertions are automatically enabled for the subpackages. Assertions can be selectively enable for the unnamed package in the current working directory. |  |
| 6 | c           | AssertionError  | An AssertionError is thrown to indicate that an assertion has failed. Don't be fooled by the name AssertionException.  |
| 7 | b<br>e      | With assertions enabled it prints 210210 followed by an AssertionError message. With assertions disabled it prints 210210-1   | If, under normal operating circumstances, the default label of a switch statement should not be reached, then an assert statement can be placed after the default label to verify that an unexpected condition has not occurred.   |
| 8 | b<br>e<br>g | With assertions enabled it prints 012. With assertions disabled it attempts to print an infinite sequence of numbers. As a rule, the boolean expression of an assert statement should not be used to perform actions that are required for normal operation of the program.                   | An assert statement should not be used as demonstrated in the program. The boolean expression of the do-loop depends on the value of the local variable i1. The value of i1 is set within the boolean expression of the assert statement. If assertions are disabled, then the boolean expression of the assert statement is not processed and the value of i1 is not updated with each iteration of the loop; so the loop runs indefinitely.  |
| 9 | a<br>d      | Assertions should not be used to validate arguments passed to public methods. Assertions should not be used to perform processing that is required for the normal operation of the application.   | Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method. Since assertions are not always enabled, an assertion should not be used to perform operations that are required for the normal operation of the program. For example, the boolean expression of an assertion should |

|    |        |   |
|----|--------|---|
|    |        | not be used to produce the side effect of incrementing a variable that controls a loop statement. If assertions are disabled then the loop is unlikely to function as intended. Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compile-time error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement.   |
| 10 | b<br>c | <p>The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification. A catch clause should not be used to catch an AssertionError.</p> <p>Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compile-time error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement. While the exception handling mechanisms of Java have been designed to allow for recovery from Exceptions, the assertion mechanisms have been designed to discourage recovery attempts. An assertion is used to verify that the program has not strayed beyond the bounds of expected behavior. For example, suppose that you go to bed one night, and your pet dog is sleeping on the floor next to your bed. Before going to sleep, you make the assertion that your dog will still be there in the morning. When you wake up, you find that a different dog is sleeping in place of your pet. How do you recover from the failure of your assertion? Since you probably did not expect your dog to be mysteriously replaced during the night, it is unlikely that you have already developed an effective recovery routine. However, if you had planned for a dog swapping exception, then the recovery should</p> |

|    |                  |  |  |
|----|------------------|--|--|
|    |                  | be handled by the exception handling mechanism rather than the assertion mechanism.  |  |
| 11 | a<br>b<br>d      | With assertions enabled it prints ABC followed by an AssertionError message. With assertions disabled it prints ABC followed by an AssertionError message. In this code example an assert statement could not be used in place of the "throw" statement. | If the default label of a switch statement should not be reached under normal operating circumstances, then the default label might be a good location for an assert statement. If a method is declared with a non-void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert, because the compiler knows that the assert statement is not functional when assertions are disabled.           |
| 12 | a<br>d           | With assertions enabled it prints ABC followed by an AssertionError message. With assertions disabled it prints ABCE   | If the default label of a switch statement should not be reached under normal operating circumstances, then the default label might be a good candidate for the use of an assert statement.  |
| 13 | d<br>e           | In this code example a throw statement must be used in place of the assert statement. Compile-time error   | If the default label of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement. If a method is declared with a non-void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert, because the compiler knows that the assert statement is not functional when assertions are disabled. |
| 14 | a<br>d<br>f      | With assertions enabled it prints an AssertionError message. With assertions disabled it prints nothing. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.      | The assert statement indicates that the programmer believes that b1 and b2 will never be true simultaneously, and the assert statement should not be reached under normal operating conditions.  |
| 15 | a<br>d<br>f<br>h | If assertions are enabled at run time it prints an error message. With assertions disabled it prints 1112. The assert statements are being used to check a precondition--something that  | Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced  |

|    |  |  |
|----|--|--|
|    | <p>must be true when the method is invoked. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method.</p>  | <p>even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method.</p>   |
| 16 | <p>b<br/>c<br/>e</p> <p>If assertions are not enabled at run time it prints nothing. With assertions enabled it prints an error message. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.</p>  | <p>Variable c equals 200 when the assertion is checked.</p>  |
| 17 | <p>b<br/>d<br/>e</p> <p>If assertions are not enabled at run time it prints nothing. With assertions enabled it prints nothing. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.</p>   | <p>This question is an example of using assertions to check a class invariant--something that must be true about each instance of the class. Although a class invariant must be true before and after the execution of each public method, the invariant is typically only checked at the end of each method and constructor.</p>  |
| 18 | <p>a<br/>d<br/>f<br/>h</p> <p>With assertions enabled it prints an error message. With assertions disabled it prints: true,true,false. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2 or b3, to be true. The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program.</p> | <p>Method m1 has a series of if/else statements. The first if statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. A set of three booleans can exist in eight states. The three if statements account for three of those states; so five more states remain. The assert statement indicates that the programmer assumes that only one of those five remaining states is valid--that is the state where only b3 is true. The combination of the three if statements and the assert statement indicate that the programmer believes that no more than one of the booleans will be true at that point in the program. That assumption is called an internal invariant.</p> |
| 19 | <p>a<br/>c<br/>e</p> <p>Prints "Only b1 is true" followed by an error message. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than</p>  | <p>Method m1 has a series of if/else statements. The first if statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. The fourth is processed if only b3 is true. A set of three booleans can exist in one of</p>  |

|    |   |   |
|----|---|---|
|    | <p>one boolean, b1, b2, or b3, to be true. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.</p>                       | <p>eight states. The first four if statements account for four of those states; so four more states remain. The combination of the three if statements and the fact that an <code>AssertionError</code> is thrown from the last else block indicates that the programmer believes that no more than one of the booleans will be true when method <code>m1</code> is being processed. An assumption concerning the state of a set of variables is called an internal invariant. In this case, however, the assertion was tested by verifying that control never reached a particular point in the program. Based on the testing technique, we would say that the assertion tests a control-flow invariant. A <code>throw</code> statement is used in place of an <code>assert</code> statement, because the <code>throw</code> statement can not be disabled. As a result, the method is certain to generate an error once control passes beyond all of the return statements. The declared return type of method <code>m1</code> is <code>String</code>. No return statement appears after the sequence of if statements; therefore, every if statement must either return a <code>String</code> or throw an exception. Assertions can be disabled at run time, so an <code>assert</code> statement in the final if block is no guarantee that an exception will be thrown. For that reason, an <code>assert</code> can not replace the <code>throw</code> statement.</p> |
| 20 | <p>b<br/>e<br/>f</p> <p>The flow of control does not reach a particular point in the program. The default case of a switch statement is not reached. The else block of an if/else statement is not reached.</p> | <p>A control-flow invariant is placed at a point in the program that the programmer assumes will never be reached. Two examples are the default case of a switch statement or the else block of an if/else statement. It makes no sense to use an <code>assert</code> statement to verify that the flow of control does reach a particular point in the program, because it is unlikely that an assertion error is helpful when the program is found to be functioning correctly. An <code>assert</code> statement placed at the beginning of a method is generally used to check a precondition. An <code>assert</code> statement that is placed at the end of a method to check the state of some variables is generally said to be checking a post condition. However, it is also possible that an <code>assert</code> statement placed at the end of a method might also be checking a control-flow invariant. The correct term</p>   |

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

depends on the usage.

## GARBAGE COLLECTIONS

### Question 1

```
class B {  
    private String name;  
    public B(String s) {name = s;}  
    protected void finalize() {System.out.print(name);}  
}  
class E {  
    public static void m() {  
        B x1 = new B("X"), y1 = new B("Y");  
    }  
    public static void main(String[] args) {  
        m(); System.gc();  
    }  
}
```

Which of the following could not be a result of attempting to compile and run the program?

- a. Prints: XY
- b. Prints: YX
- c. Prints: XXYY
- d. Nothing is printed.
- e. None of the above

### Question 2

```
void m1() {  
    Q q1 = null;  
    for (int i = 0; i < 10; i++) {  
        q1 = new Q(); // 1  
        m2(q1); // 2  
    }  
    System.out.print("All done"); // 3  
}
```

When the processing of line 3 begins, how many objects of type Q that were created at line 1 have become eligible for garbage collection?

- a. 0
- b. 1
- c. 9
- d. 10

- e. Indeterminate.
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 3

```
class Q {  
    private int id;  
    protected void finalize() {System.out.print(id);}  
    public Q(int i) {id = i;}  
}  
class R {  
    public static void main(String[] args) {  
        Q q1 = null;  
        for (int i = 0; i < 10; i++) {q1 = new Q(i);} // 1  
        System.gc(); // 2  
    }  
}
```

When the processing of line 2 begins, how many objects of type Q that were created at line 1 have become eligible for garbage collection?

- a. 0
- b. 1
- c. 9
- d. 10
- e. Indeterminate.
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 4

```
class I {  
    private I other;  
    public void other(I i) {other = i;}  
}  
class J {  
    private void m1() {  
        I i1 = new I(), i2 = new I();  
        I i3 = new I(), i4 = new I();  
        i1.other(i3); i2.other(i1);  
        i3.other(i2); i4.other(i4);  
    }  
    public static void main (String[] args) {  
        new J().m1();  
    }  
}
```

}}

Which object is not eligible for garbage collection after method m1 returns?

- a. i1
- b. i2
- c. i3
- d. i4
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 5

```
class I {  
    private String name;  
    public I(String s) {name = s;}  
    private I other;  
    public void other(I i) {other = i;}  
}  
class J {  
    private I i1 = new I("A"), i2 = new I("B"), i3 = new I("C");  
    private void m1() {  
        i1.other(i2); i2.other(i1); i3.other(i3);  
        i1 = i3; i2 = i3;  
        m2();  
    }  
    private void m2() /* Do amazing things. */{  
    }  
    public static void main (String[] args) {  
        new J().m1();  
    }  
}
```

Which of the three objects, A, B or C, is not eligible for garbage collection when method m2 begins to execute?

- a. A
- b. B
- c. C
- d. None of the above

### Question 6

```
class I {  
    private String name;  
    protected void finalize() {System.out.print(name);}   
    public I(String s) {name = s;}  
}
```

```

class J {
    private static void m1(I[] a1) {
        a1[0] = a1[1] = a1[2] = null;
    }
    public static void main (String[] args) {
        I[] a1 = new I[3]; // 1
        a1[0] = new I("A"); // 2
        a1[1] = new I("B"); // 3
        a1[2] = new I("C"); // 4
        m1(a1);
        System.gc();
    }
}

```

After method m1 returns, the object created on which line is not eligible for garbage collection?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above
- f. Compile-time error
- g. Run-time error

### Question 7

```

class I {
    private String name;
    public String toString() {return name;}
    public I(String s) {name = s;}
}
class J {
    private static void m1(I[] a1) {a1 = null;}
    public static void main (String[] args) {
        I[] a1 = new I[3]; // 1
        a1[0] = new I("A"); // 2
        a1[1] = new I("B"); // 3
        a1[2] = new I("C"); // 4
        m1(a1);
        for (int i = 0; i < a1.length; i++) {
            System.out.print(a1[i]);
        }
    }
}

```

After method m1 returns, the object created on which line is eligible for garbage collection?

- a. 1
- b. 2

- c. 3
- d. 4
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 8

```
class A {  
    private String name;  
    private A otherA;  
    public A(String name) {this.name = name;}  
    public void other(A otherA) {this.otherA = otherA;}  
    public A other() {return otherA;}  
    public String toString() {return name;}  
    protected void finalize() {System.out.print(name);} }  
class B {  
    public static void m1() {  
        A a1 = new A("A1"), a2 = new A("A2"), a3 = new A("A3"), a0 = a3;  
        a1.other(a2); a2.other(a3); a3.other(a1);  
        for(int i = 0; i<4; i++){System.out.print(a0 = a0.other());}  
    }  
    public static void main(String[] args) {m1(); System.gc();}  
}
```

Which of the following could be a result of attempting to compile and run the program?

- a. A1A2A3A1
- b. A0A0A0A0A1A2A3
- c. A1A2A3A1A2A3
- d. A1A2A3A1A1A2A3
- e. A1A2A3A1A3A2A1
- f. A0A1A2A3A1A2A3

### Question 9

```
class B {  
    private String name;  
    public B(String name) {this.name = name;}  
    public String toString() {return name;}  
    protected void finalize() {System.out.print(name);} }  
class H {  
    static B ba = new B("Ba");  
    static int i = 1;
```

```

static B m1(B b) {return b = new B("B" + i++);}
public static void main (String[] args) {
    B x = m1(ba); m1(x);
    System.out.println(", " + ba + ", " + x);
}

```

Which of the following could be a result of attempting to compile and run the program?

- a. Ba, B1, B2
- b. B1, Ba, B2
- c. , Ba, B1
- d. B2, Ba, B1
- e. BaB1b2, null, null
- f. B1B2, ba, null

### Question 10

```

class B {
    private String name;
    public B(String name) {this.name = name;}
    public String toString() {return name;}
    protected void finalize() {System.out.print(name);}
}
class J {
    static B bc;
    static int i = 1;
    static B m1(B b) {bc = b; return new B("B" + i++);}
    public static void main (String[] args) {
        B x = m1(new B("Ba")), y = m1(new B("Bb"));
        System.out.println(", " + x + ", " + y + ", " + bc);
    }
}

```

Which of the following could be a result of attempting to compile and run the program?

- a. BaBb, B1, B2, B2
- b. B1B2, null, null, Bb
- c. , Ba, Bb, Bb
- d. BaBbB1B2, null, null, null
- e. Ba, B1, B2, Bb
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 11

```

class I {
    private String name;

```

```

public String name() {return name;}
public I(String s) {name = s;}
}
class J {
public static void m1(I i) {i = null;}
public static void main (String[] args) {
    I i = new I("X");          // 1
    m1(i);                  // 2
    System.out.print(i.name()); // 3
}
}

```

Which of the following is a true statement?

- a. The object created at line 1 is eligible for garbage collection after line 2.
- b. A NullPointerException is generated at line 3.
- c. The program compiles, runs and prints X.
- d. The program fails to compile.
- e. None of the above

| No. | Answer           | Remark  |
|-----|------------------|---|
| 1   | c Prints: XXYY   | The program will not print XXYY. Please note that the question asks which could NOT be a result of attempting to compile and run the program. The finalize method of each instance can only run once; so X or Y can never be printed more than once. The instances referenced by x1 and y1 become eligible for garbage collection when method m returns; so both could be finalized at that time, but there is no guarantee that they will be. Even though System.gc is invoked in the main method, there is no guarantee that the garbage collector will run at that time. If the garbage collector does run before the program terminates, then the name of each object could be printed at most one time. The order in which the names are printed depends on the order in which the objects are finalized. If the garbage collector does not run, then nothing will be printed. |
| 2   | e Indeterminate. | Since we don't know what method m2 might be doing, we can not know if the objects are eligible for garbage collection. Suppose that method m2 is declared inside of a class that also contains 10 instance variables (instance variables are non-static member fields) that are references to instances of class A. The argument that appears in the method invocation expression m2(q1) is a reference to an instance of class Q. Suppose that m2 saves each argument value in one of the ten instance variables or in an element of an array of type Q[]. When the  |

|   |   |                   |   |
|---|---|-------------------|---|
|   |   |                   | loop in method m1 runs to completion, each instance of class Q would still be referenced by one of the ten instance variables. Since the instance variables would continue to reference each instance of class Q when line 3 is executed, none of the instances would be eligible for garbage collection at that point. A second possibility is that method m2 does not save the reference values. In that case, all of the instances that were created inside the loop would be eligible for garbage collection when line 3 is executed.   |
| 3 | c | 9                 | With each pass through the loop, q1 references a new object, and the old object becomes eligible for garbage collection. When the processing of line 2 begins, the last object referenced by q1 is not eligible for garbage collection.   |
| 4 | g | None of the above | Please note that this question asks which object is NOT eligible for garbage collection after method m1 returns. The objects referenced by i1, i2 and i3 form a ring such that each object is referenced by another. Even so, nothing outside of method J.m1 references any of those objects. When method J.m1 returns, the ring becomes an island of isolated objects that are not reachable by any part of the user program. A key point to remember is that an object that is referenced by another object can be eligible for garbage collection if the two objects form an island of isolated objects. |
| 5 | c | C                 | Please note that this question asks which objects are NOT eligible for garbage collection when method m2 begins to execute? All three references, i1, i2 and i3, refer to object named C; so C is not eligible for garbage collection when method m2 begins to execute. The objects named A and B have references to each other, but no other objects refer to A and B. The objects A and B form an island of isolated objects and are eligible for garbage collection.   |
| 6 | a | 1                 | Please note that this question asks which objects are NOT eligible for garbage collection after method m1 returns. After method m1 returns, the array a1 created on line 1 is not eligible for garbage collection. Method m1 sets all elements of the array to null; so the objects created on lines 2, 3 and 4 are eligible for garbage collection when method m1 returns.   |
| 7 | g | None of the above | After method m1 returns, none of the objects are eligible for garbage collection. Method m1 sets the  |

|   |  |  |
|---|--|--|
|   |  | parameter variable a1 to null, but that does not change the reference a1 in the J.main method. Since array a1 continues to reference all three objects, none of the three are eligible for garbage collection.   |
| 8 | a A1A2A3A1<br>c A1A2A3A1A2A3<br>d A1A2A3A1A1A2A3<br>e A1A2A3A1A3A2A1 | The three instances of class A form an isolated ring where each instance references the next instance and the third references the first instance. Four iterations of the for loop are processed. Inside the body of the for statement, the invocation of the print method contains the argument expression a0 = a0.other(). On the first iteration, the reference variable a0 references the instance named A3. The value returned by the method named other is a reference to the instance named A1. The reference is assigned to the reference variable a0 and is also the value produced by the expression a0 = a0.other(). That reference value is passed as an argument to the print method, and the print method invokes the A.toString method. With each iteration of the loop, the reference moves to the next object in the loop and the name of the object is printed. After four iterations, the loop ends and the method m1 returns. The invocation of the System.gc method serves as a suggestion that the garbage collector should be allowed to run. The system could ignore the suggestion, so there is no guarantee that the eligible arguments will be garbage collected. If they are collected, there is no guarantee which will be collected first. The only guarantee is that the finalize method will be invoked on each particular instance before the resources that had been allocated to that instance are reclaimed. |
| 9 | c , Ba, B1<br>d , B2, Ba, B1   | Class H declares two static member variables named ba and i. The type of i is int, and the value is initialized to 1. The type of ba is B. The declaration of ba contains the class instance creation expression new B("Ba"). The constructor of class B assigns the argument value to the instance variable called name. Inside the main method of class H, the method invocation expression m1(ba) invokes method m1. The argument is the static member variable ba. The body of method m1 contains a return statement with the expression b = new B("B" + i++). The assignment expression contains the class instance creation expression new B("B" + i++) which creates  |

a new instance of the class B. For this first invocation of method m1, the argument appearing in the class instance creation expression is the String value B1. The reference to the new String is assigned to the parameter variable b, but that assignment does not change the value of the member variable ba. The value of the assignment expression is the reference to the new instance of class B with the name B1, and that reference value is returned by the method m1. The returned value is assigned to the local variable x. The next statement inside the main method is another invocation of method m1. The argument appearing in the method invocation expression m1(x) is the local reference variable x. The method invocation does not change the value of x. The value returned by this second invocation of m1 is a reference to a new instance of class B that has the name B2. The returned reference value is not assigned to a variable, so the instance named B2 is eligible for garbage collection. There is no guarantee that the garbage collector will run before the print statement is invoked. If it does run, then the instance named B2 could be finalized causing the name to be printed.

Class J declares two static member variables named bc and i. The type of i is int, and the value is initialized to 1. The type of bc is B. Inside the main method of class J, the method invocation expression m1(new B("Ba")) invokes method m1. The argument is the class instance creation expression new B("Ba"). The constructor of class B assigns the argument value to the instance variable called name, so a new instance of class B named Ba is created. The reference to the new instance of class B is the argument that is passed to method m1. The body of method m1 contains two statements. The first contains the assignment expression bc = b that assigns the value of the method parameter b to the static member variable bc, so bc now references the instance of class B named Ba. The second statement in the body of m1 is a return statement with the class instance creation expression new B("B" + i++). For this first invocation of method m1, the argument appearing in the class instance creation expression is the String value B1. The reference to the new String is returned by method m1. The returned value is

10 e Ba, B1, B2, Bb

|    |   |  |  |
|----|---|--|--|
|    |   |  | assigned to the local variable x. Inside the main method, the declaration of the local variable y contains another invocation of method m1. The argument appearing in the method invocation expression m1(new B("Bb")) is the class instance creation expression new B("Bb"), so the argument value for the invocation of method m1 is a reference to a new instance of class B named Bb. Inside the body of method m1, the reference to the new instance of class B named Bb is assigned to the static member variable Bc. At that point, the instance of class B named Ba becomes eligible for garbage collection. Method m1 returns a reference to a new instance of class B named B2. There is no guarantee that the garbage collector will run before the print statement is invoked. If it does run, then the instance named Ba could be finalized causing the name to be printed. |
| 11 | c | The program compiles, runs and prints X. | The parameter i of method m1 is a copy of the local variable i of method J.main. Setting the parameter variable i of method m1 to null does not change the local variable i of method J.main.  |

## THREADS

### Question 1

Which of the following methods are members of the Object class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 2

Which of the following methods are static members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 3

Which of the following methods are deprecated members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. resume
- e. run
- f. sleep
- g. start
- h. stop
- i. suspend
- j. yield
- k. wait

### Question 4

Which of the following methods name the InterruptedException in its throws clause?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 5

A timeout argument can be passed to which of the following methods?

- a. join
- b. notify
- c. notifyAll

- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 6

Which of the following instance methods should only be called by a thread that holds the lock of the instance on which the method is invoked?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. start
- f. wait

### Question 7

Which of the following is a checked exception?

- a. IllegalMonitorStateException
- b. IllegalThreadStateException
- c. IllegalArgumentException
- d. InterruptedException
- e. None of the above

### Question 8

Which kind of variable would you prefer to synchronize on?

- a. A member variable of a primitive type
- b. A member variable that is an object reference
- c. A method local variable that is a reference to an instance that is created within the method
- d. None of the above

### Question 9

synchronized (expression) block

The synchronized statement has the form shown above. Which of the following are true statements?

- a. A compile-time error occurs if the expression produces a value of any reference type

- b. A compile-time error occurs if the expression produces a value of any primitive type
- c. A compile-time error does not occur if the expression is of type boolean
- d. The synchronized block may be processed normally if the expression is null
- e. If execution of the block completes normally, then the lock is released
- f. If execution of the block completes abruptly, then the lock is released
- g. A thread can hold more than one lock at a time
- h. Synchronized statements can be nested
- i. Synchronized statements with identical expressions can be nested

### Question 10

Which of the following is a true statement?

- a. The process of executing a synchronized method requires the thread to acquire a lock
- b. Any overriding method of a synchronized method is implicitly synchronized
- c. If any method in a class is synchronized, then the class itself must also be declared using the synchronized modifier
- d. If a thread invokes a static synchronized method on an instance of class A, then the thread must acquire the lock of that instance of class A
- e. None of the above

### Question 11

Which of the following thread state transitions model the lifecycle of a thread?

- a. The Dead state to the Ready state
- b. The Ready state to the Not-Runnable state
- c. The Ready state to the Running state
- d. The Running state to the Not-Runnable state
- e. The Running state to the Ready state
- f. The Not-Runnable state to the Ready state
- g. The Not-Runnable state to the Running state

### Question 12

Which of the following are true statements?

- a. The Thread.yield method might cause the thread to move to the Not-Runnable state
- b. The Thread.yield method might cause the thread to move to the Ready state
- c. The same thread might continue to run after calling the Thread.yield method
- d. The Thread.yield method is a static method
- e. The behavior of the Thread.yield method is consistent from one platform to the next
- f. The Thread.sleep method causes the thread to move to the Not-Runnable state
- g. The Thread.sleep method causes the thread to move to the Ready state

### **Question 13**

Which of the following will not force a thread to move into the Not-Runnable state?

- a. Thread.yield method
- b. Thread.sleep method
- c. Thread.join method
- d. Object.wait method
- e. By blocking on I/O
- f. Unsuccessfully attempting to acquire the lock of an object
- g. None of the above

### **Question 14**

Which of the following will cause a dead thread to restart?

- a. Thread.yield method
- b. Thread.join method
- c. Thread.start method
- d. Thread.resume method
- e. None of the above

### **Question 15**

When a thread is created and started, what is its initial state?

- a. New
- b. Ready
- c. Not-Runnable
- d. Running
- e. Dead
- f. None of the above

### **Question 16**

Which of the following are true statements?

- a. The Thread.run method is used to start a new thread running
- b. The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler
- c. The Runnable interface declares the start method
- d. The Runnable interface declares the run method
- e. The Thread class implements the Runnable interface
- f. If an Object.notify method call appears in a synchronized block, then it must be the last method call in the block
- g. No restriction is placed on the number of threads that can enter a synchronized method

- h. Some implementations of the Thread.yield method will not yield to a thread of lower priority

### Question 17

Which of the following are true statements?

- a. Thread.MAX\_PRIORITY == 10
- b. Thread.MAX\_PRIORITY == 5
- c. Thread.NORM\_PRIORITY == 5
- d. Thread.NORM\_PRIORITY == 3
- e. Thread.NORM\_PRIORITY == 0
- f. Thread.MIN\_PRIORITY == 1
- g. Thread.MIN\_PRIORITY == 0
- h. Thread.MIN\_PRIORITY == -5
- i. Thread.MIN\_PRIORITY == -10

### Question 18

Which of the following are true statements?

- a. A program will terminate only when all daemon threads stop running
- b. A program will terminate only when all user threads stop running
- c. A daemon thread always runs at Thread.MIN\_PRIORITY
- d. A thread inherits its daemon status from the thread that created it
- e. The daemon status of a thread can be changed at any time using the Thread.setDaemon method
- f. The Thread.setDaemon method accepts one of two argument values defined by the constants Thread.DAEMON and Thread.USER

### Question 19

```
class A extends Thread {  
    public A(Runnable r) {super(r);}  
    public void run() {System.out.print("A");}  
}  
class B implements Runnable {  
    public void run() {System.out.print("B");}  
}  
class C {  
    public static void main(String[] args) {  
        new A(new B()).start();  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 20

```
class A implements Runnable {  
    public void run() {System.out.print(Thread.currentThread().getName());}  
}  
class B implements Runnable {  
    public void run() {  
        new A().run();  
        new Thread(new A(),"T2").run();  
        new Thread(new A(),"T3").start();  
    }  
}  
class C {  
    public static void main (String[] args) {  
        new Thread(new B(),"T1").start();  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: T1T1T1
- b. Prints: T1T1T2
- c. Prints: T1T2T2
- d. Prints: T1T2T3
- e. Prints: T1T1T3
- f. Prints: T1T3T3
- g. Compile-time error
- h. Run-time error
- i. None of the above

### Question 21

```
class AnException extends Exception {}  
class A extends Thread {  
    public void run() throws AnException {  
        System.out.print("A"); throw new AnException();  
    }  
}  
class B {  
    public static void main (String[] args) {  
        A a = new A(); a.start(); System.out.print("B");  
    }  
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 22

```
class A extends Thread {  
    public void run() {System.out.print("A");}  
}  
class B {  
    public static void main (String[] args) {  
        A a = new A();  
        a.start();  
        a.start(); // 1  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error
- b. The second attempt to start thread t1 is successful
- c. The second attempt to start thread t1 is ignored
- d. Compile-time error at marker 1
- e. An IllegalThreadStateException is thrown at run-time
- f. None of the above

| No. | Answer                            | Remark  |
|-----|-----------------------------------|---|
| 1   | b<br>c notify notifyAll wait<br>h |   |
| 2   | e sleep yield<br>g                |   |
| 3   | d<br>h resume stop suspend<br>i   | For the purposes of the exam, you don't need to memorize the deprecated methods of the Thread class. Even though a question such as this will not be on the exam, every Java programmer should know that the deprecated methods should not be used in new programs. |

|    |  |   |  |
|----|--|---|--|
|    | a  |   |  |
| 4  | e<br>join<br>sleep<br>wait<br>h  |   |  |
| 5  | a<br>e<br>join<br>sleep<br>wait<br>h   |   |  |
| 6  | b<br>c<br>notify<br>notifyAll<br>wait<br>f   |   |  |
| 7  | d<br>InterruptedException  | The methods Object.wait, Thread.join and Thread.sleep name InterruptedException in their throws clauses.  |  |
| 8  | b<br>A member variable that is an object reference   | Primitives don't have locks; therefore, they can not be used to synchronize threads. A method local variable that is a reference to an instance that is created within the method should not be used to synchronize threads, because each thread has its own instance of the object and lock. Synchronization on an instance that is created locally makes about as much sense as placing on your doorstep a box full of keys to the door. Each person that comes to your door would have their own copy of the key; so the lock would provide no security. |  |
| 9  | b<br>A compile-time error occurs if the expression produces a value of any primitive type. If execution of the block completes normally, then the lock is released. If execution of the block completes abruptly, then the lock is released.<br>e<br>f<br>g<br>h<br>i<br>A thread can hold more than one lock at a time. Synchronized statements can be nested. Synchronized statements with identical expressions can be nested |   |  |
| 10 | a<br>The process of executing a synchronized method requires the thread to acquire a lock  | The synchronized modifier can not be applied to a class. A method that overrides a synchronized method does not have to be synchronized. If a thread invokes a synchronized instance method on an instance of class A, then the thread must acquire the lock of that instance of class A. The same is not true for  |  |

|    |                  |   |   |
|----|------------------|---|---|
|    |                  |   | synchronized static methods. A synchronized static method is synchronized on the lock for the Class object that represents the class for which the method is a member.  |
| 11 | c<br>d<br>e<br>f | The Ready state to the Running state<br>The Running state to the Not-Runnable state<br>The Running state to the Ready state<br>The Not-Runnable state to the Ready state  | A dead thread can not be restarted.   |
| 12 | b<br>c<br>d<br>f | The Thread.yield method might cause the thread to move to the Ready state<br>The same thread might continue to run after calling the Thread.yield method<br>The Thread.yield method is a static method<br>The Thread.sleep method causes the thread to move to the Not-Runnable state                             | The Thread.yield method is intended to cause the currently executing thread to move from the Running state to the Ready state and offer the thread scheduler an opportunity to allow a different thread to execute based on the discretion of the thread scheduler. The thread scheduler may select the same thread to run immediately, or it may allow a different thread to run. The Thread.yield method is a native method; so the behavior is not guaranteed to be the same on every platform. However, at least some implementations of the yield method will not yield to a thread that has a lower priority. |
| 13 | a                | Thread.yield method   | The Thread.yield method may cause a thread to move into the Ready state, but that state transition is not guaranteed. The JLS states that the Thread.yield method provides a hint to the thread scheduler, but the scheduler is free to interpret--or ignore--the hint as it sees fit. Nothing in the JLS suggests that the thread might move to the Not-Runnable state.  |
| 14 | e                | None of the above   | A dead thread can not be restarted.   |
| 15 | b                | Ready   |   |
| 16 | b<br>d<br>e<br>h | The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler<br>The Runnable interface declares the run method<br>The Thread class implements the Runnable interface<br>Some implementations of the Thread.yield method will not yield to a thread of lower priority | The Object.notify method can only be called by the thread that holds the lock of the object on which the method is invoked. Suppose that thread T1 enters a block that is synchronized on an object, A. Within the block, thread T1 holds the lock of A. Even if thread T1 calls the notify method immediately after entering the synchronized block, no other thread   |

|    |   |  |  |
|----|---|--|--|
|    |   |  | can grab the lock of object A until T1 leaves the synchronized block. For that reason, the transfer of control from thread T1 to any waiting thread can not be accelerated by moving the notify method to an earlier point in the synchronized block. The behavior of Thread.yield is platform specific. However, at least some implementations of the yield method will not yield to a thread that has a lower priority. Invoking the Thread.yield method is like offering a suggestion to the JVM to allow another thread to run. The response to the suggestion is platform specific. |
| 17 | a Thread.MAX_PRIORITY == 10<br>c Thread.NORM_PRIORITY == 5<br>f Thread.MIN_PRIORITY == 1  |  |  |
| 18 | A program will terminate only when all<br>b user threads stop running A thread<br>d inherits its daemon status from the<br>thread that created it |  |  |
| 19 | a Prints: A   |  | If a Runnable target object is passed to the constructor of the Thread class, then the Thread.run method will invoke the run method of the Runnable target. In this case, the Thread.run method is overridden by A.run. The A.run method does nothing more than print the letter A. The invocation of the A.start method inside the main method results in the invocation of A.run, and the letter A is printed. The B.run method is never invoked.  |
| 20 | e Prints: T1T1T3  |  | The Thread.currentThread method returns a reference to the currently executing thread. When the run method is invoked directly it does not start a new thread; so T1 is printed twice.   |
| 21 | e Compile-time error  |  | The Runnable.run method does not have a throws clause; so any implementation of run can not throw a checked exception.   |
| 22 | e An IllegalThreadStateException is thrown at run-time  |  | For the purposes of the exam, invoking the start method on a thread that has already been started will generate an   |

IllegalThreadStateException. The actual behavior of the method might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. For the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification and ignores the actual behavior of the 1.4 version of the JVM.

### Question 1

```
class A extends Thread {  
    private int i;  
    public void run() {i = 1;}  
    public static void main(String[] args) {  
        A a = new A(); a.start(); System.out.print(a.i);  
    }  
}
```

What are the possible results of attempting to compile and run the program?

- a. Prints nothing
- b. Prints: 0
- c. Prints: 1
- d. Prints: 01
- e. Prints: 10
- f. Compile-time error
- g. Run-time error

### Question 2

```
class A extends Thread {  
    private int i;  
    public void run() {i = 1;}  
    public static void main(String[] args) {  
        A a = new A(); a.run(); System.out.print(a.i);  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing
- b. Prints: 0
- c. Prints: 1
- d. Prints: 01
- e. Prints: 10
- f. Compile-time error
- g. Run-time error
- h. None of the above

### Question 3

```
class A extends Thread {  
    public void run() {  
        try {sleep(10000);} catch (InterruptedException ie){}  
    }  
    public static void main(String[] args) {  
        A a1 = new A();  
        long startTime = System.currentTimeMillis();  
        a1.start();  
        System.out.print(System.currentTimeMillis() - startTime);  
    }  
}
```

What are the possible results of attempting to compile and run the program?

- a. Prints a number greater than or equal to 0
- b. The number printed must always be greater than 10000
- c. This program will run for at least ten seconds
- d. Compile-time error
- e. Run-time error

### Question 4

Which of the following is used to force each thread to reconcile its working copy of a variable with the master copy in main memory?

- a. final
- b. static
- c. synchronized
- d. transient
- e. volatile
- f. native

### Question 5

```

class A extends Thread {
    public void run() {
        synchronized (this) {
            try {wait(5000);} catch (InterruptedException ie){}
        }
    }
    public static void main(String[] args) {
        A a1 = new A();
        long startTime = System.currentTimeMillis();
        a1.start();
        System.out.print(System.currentTimeMillis() - startTime + ",");
        try {a1.join(6000);} catch (InterruptedException ie) {}
        System.out.print(System.currentTimeMillis() - startTime);
    }
}

```

What are the possible results of attempting to compile and run the program?

- a. The first number printed is greater than or equal to 0
- b. The first number printed must always be greater than 5000
- c. The second number printed must always be greater than 5000
- d. The second number printed must always be greater than 6000
- e. The synchronized block inside the run method is not necessary
- f. Compile-time error
- g. Run-time error

### Question 6

```

class A extends Thread {
    String[] sa;
    public A(String[] sa){this.sa = sa;}
    public void run() {
        synchronized (sa) {System.out.print(sa[0] + sa[1] + sa[2]);}
    }
}
class B {
    private static String[] sa = new String[]{"X","Y","Z"};
    public static void main (String[] args) {
        synchronized (sa) {
            Thread t1 = new A(sa); t1.start();
            sa[0] = "A"; sa[1] = "B"; sa[2] = "C";
        }
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: XYZ
- b. Prints: AYZ
- c. Prints: ABZ
- d. Prints: ABC
- e. Compile-time error
- f. Run-time error

- g. None of the above

### Question 7

```
class A extends Thread {  
    String[] sa;  
    public A(String[] sa) {this.sa = sa;}  
    public void run() {  
        synchronized (sa) {  
            while (!sa[0].equals("Done")) {  
                try {sa.wait();} catch (InterruptedException ie) {}  
            }  
            System.out.print(sa[1] + sa[2] + sa[3]);  
        }  
    }  
}  
class B {  
    private static String[] sa = new String[]{"Not Done","X","Y","Z"};  
    public static void main (String[] args) {  
        Thread t1 = new A(sa); t1.start();  
        synchronized (sa) {  
            sa[0] = "Done";  
            sa[1] = "A"; sa[2] = "B"; sa[3] = "C";  
            sa.notify();  
        }  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: XYZ
- b. Prints: AYZ
- c. Prints: ABZ
- d. Prints: ABC
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 8

Which of the following are true statements?

- a. The Thread.join method is static
- b. The Thread.join method is always invoked on an instance of Thread
- c. The Thread.join method causes the current thread to wait for the referenced thread to die
- d. The Thread.join method declares an InterruptedException in the throws clause
- e. The Thread.join method accepts a timeout value as an argument
- f. The timeout value sets the minimum time that the current thread will wait for the death of the referenced thread

- g. Thread.join will return immediately if the timeout value is zero
- h. A timeout of zero will allow Thread.join to wait forever if necessary

### Question 9

Which of the following allows a thread t1 to become the holder of the lock of object obj1.

- a. By blocking on I/O
- b. By entering a synchronized instance method of the obj1
- c. By invoking the wait method on the object
- d. By entering the body of a block that is synchronized on obj1
- e. By entering a synchronized static method of the obj1
- f. By invoking the notify method on obj1

### Question 10

After invoking the wait method on an object, obj1, a thread, T1, will remain in the wait set of obj1 until which of the following occurs?

- a. Another thread invokes the notify method on the object, obj1, and T1 is selected to move out of the wait set
- b. Another thread invokes the notifyAll method on the object
- c. Another thread invokes the resume method on thread T1
- d. Another thread interrupts thread T1
- e. The priority of thread T1 is increased
- f. A specified timeout period has elapsed
- g. Another thread invokes the join method on thread T1

### Question 11

```
class A implements Runnable{public void run() {}}
class B {
    public static void main(String[] args) {
        Thread t1 = new Thread();          // 1
        Thread t2 = new Thread(new A());    // 2
        Thread t3 = new Thread(new A(), "A"); // 3
        Thread t4 = new Thread("A");       // 4
    }
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

## Question 12

```
class A implements Runnable{public void run() {}}  
class B {  
    public static void main(String[] args) {  
        Thread t1 = new Thread();          // 1  
        Thread t2 = new Thread(new A());    // 2  
        Thread t3 = new Thread("A", new A()); // 3  
        Thread t4 = new Thread("A");       // 4  
    } }
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

## Question 13

```
class A extends Thread {  
    private boolean done;  
    public void setDone(boolean done) {this.done = done;}  
    public void run() {  
        synchronized (this) {  
            while (!done) {try {wait();} catch (InterruptedException ie){}}  
        }  
    }  
    public static void main(String[] args) {  
        A a1 = new A();  
        long startTime = System.currentTimeMillis();  
        a1.start();  
        System.out.print(System.currentTimeMillis() - startTime);  
    } }
```

Which is a possible result of attempting to compile and run the program?

- a. The number printed is greater than or equal to 0
- b. The synchronized block inside the run method is not necessary
- c. This program runs to completion after the elapsed time is printed
- d. Compile-time error
- e. Run-time error
- f. None of the above

### Question 14

```
class A extends Thread {  
    public void run() {  
        synchronized (this) {  
            try {wait();} catch (InterruptedException ie){}  
        }  
    }  
    public static void main(String[] args) {  
        A a1 = new A(); a1.setDaemon(true);  
        long startTime = System.currentTimeMillis();  
        a1.start();  
        System.out.print(System.currentTimeMillis() - startTime + ",");  
    }  
}
```

Which is a possible result of attempting to compile and run the program?

- a. The number printed is greater than or equal to 0
- b. The synchronized block inside the run method is not necessary
- c. Thread a1 waits forever and the program runs forever
- d. Compile-time error
- e. Run-time error
- f. None of the above

### Question 15

```
class A extends Thread {  
    private Object obj;  
    public A(Object obj) {this.obj = obj;}  
    public void run() {  
        try {  
            synchronized (obj) {obj.wait();}  
        } catch (InterruptedException ie) {}  
        System.out.print(Thread.currentThread().getName());  
    }  
}  
class B {  
    private void m1() {  
        for (int i = 0; i < 10; i++) {  
            A t1 = new A(this);  
            t1.setName(String.valueOf(i)); t1.setDaemon(true); t1.start();  
        }  
        synchronized (this) {notifyAll();}  
    }  
    public static void main(String[] args) {new B().m1();}  
}
```

What are the possible results of attempting to compile and run the program?

- a. All of the numbers 0 through 9 must always be printed

- b. Some or all of the numbers 0 through 9 could be printed
- c. Nothing is printed
- d. Run-time error

### Question 16

```
class C extends Thread {  
    private static String[] sa = new String[]{"Not Done","X","Y","Z"};  
    public void run() {  
        synchronized (sa) {  
            while (!sa[0].equals("Done")) {  
                try {sa.wait();} catch (InterruptedException ie) {}  
            }  
            System.out.print(sa[1] + sa[2] + sa[3]);  
        }  
    }  
    public static void main (String[] args) {  
        start();  
        synchronized (sa) {  
            sa[0] = "Done";  
            sa[1] = "A"; sa[2] = "B"; sa[3] = "C";  
            sa.notify();  
        }  
    }  
}
```

Which is a possible result of attempting to compile and run the program?

- a. Prints: XYZ
- b. Prints: AYZ
- c. Prints: ABZ
- d. Prints: ABC
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 17

```
class C extends Thread {  
    private static String[] sa = new String[]{"Not Done","X","Y","Z"};  
    public void run() {  
        synchronized (this) {  
            while (!sa[0].equals("Done")) {  
                try {wait();} catch (InterruptedException ie) {}  
            }  
            System.out.print(sa[1] + sa[2] + sa[3]);  
        }  
    }  
    void m1() {  
        start();  
    }  
}
```

```

synchronized (this) {
    sa[0] = "Done";
    sa[1] = "A"; sa[2] = "B"; sa[3] = "C";
}
public static void main (String[] args) {
    new C().m1(); notify();
}

```

Which is a possible result of attempting to compile and run the program?

- a. Prints: XYZ
- b. Prints: AYZ
- c. Prints: ABZ
- d. Prints: ABC
- e. Compile-time error
- f. Run-time error
- g. None of the above

### Question 18

```

class A extends Thread {
    public void run() {
        long startTime = System.currentTimeMillis();
        long endTime = startTime + 10000;
        while (System.currentTimeMillis() < endTime) {
            yield();
        }
    }
    public static void main(String[] args) {
        A a1 = new A();
        long startTime = System.currentTimeMillis();
        a1.start(); sleep(1000); a1.interrupt(); a1.join();
        System.out.print(System.currentTimeMillis() - startTime);
    }
}

```

Which is a possible result of attempting to compile and run the program?

- a. Prints a number that is less than 1000
- b. Prints a number between 1000 and 9999
- c. Prints a number larger than 10000
- d. Compile-time error
- e. Run-time error
- f. None of the above

### Question 19

```

class A extends Thread {
    public void run() {System.out.print("A");}
}

```

```

    }
class B {
    public static void main (String[] args) {
        A a = new A(); a.start();
        try {
            a.join();
        } catch (InterruptedException ie) {ie.printStackTrace();}
        a.start(); // 1
    }
}

```

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error
- b. The second attempt to start thread t1 is successful
- c. The second attempt to start thread t1 is ignored
- d. Compile-time error at marker 1
- e. An IllegalThreadStateException is thrown at run-time
- f. None of the above

### Question 20

```

class A extends Thread {
    private static B b = new B();
    private String s1;
    public void run() {System.out.print(b.m1(s1));}
    A(String threadName, String s1) {
        super(threadName); this.s1 = s1;
    }
    public static void main (String[] args) {
        A a = new A("T1","A"), b = new A("T2","B"); a.start(); b.start();
    }
}
class B {
    private String s1;
    public synchronized String m1(String s) {
        s1 = s;
        try {Thread.sleep(1);} catch (InterruptedException ie) {}
        return "["+Thread.currentThread().getName()+","+s1+"]";
    }
}

```

What are the possible results of attempting to compile and run the program?

- a. Prints nothing
- b. Prints: [T1,A][T2,B]
- c. Prints: [T1,B][T2,B]
- d. Prints: [T2,B][T1,A]
- e. Prints: [T2,A][T1,A]
- f. Compile-time error
- g. Run-time error

## Question 21

```
class A extends Thread {  
    static long startTime;  
    public void run() {  
        for (int i = 0; i < 99999; i++) {Math.sin(i);}  
        String name = Thread.currentThread().getName();  
        long time = System.currentTimeMillis();  
        System.out.println(name + " done at " + (time - startTime));  
    }  
    public static void main(String[] args) {  
        A t1 = new A(); A t2 = new A();  
        t1.setName("T1"); t2.setName("T2");  
        t1.setPriority(Thread.MIN_PRIORITY);  
        t2.setPriority(Thread.MAX_PRIORITY);  
        startTime = System.currentTimeMillis();  
        t1.start(); t2.start();  
    }  
}
```

Which of the following is a true statement?

- a. The priority assigned to thread T2 is greater than the priority assigned to T1
- b. Java guarantees that thread T2 will get more CPU time than T1
- c. Java guarantees that thread T2 will run to completion before T1
- d. None of the above

| No. | Answer                                     | Remark   |
|-----|--|--|
| 1   | b<br>c Prints: 0 Prints: 1                 | The new thread is started before the print statement, but there is no guarantee that the new thread will run before the print statement is processed. The guarantee could be provided by placing the method invocation expression a.join() before the print statement, but the invocation of the join method does not appear in the program. If the new thread runs before the print statement is processed, then 1 is printed. Otherwise, 0 is printed. |
| 2   | c Prints: 1                                | The a.run() method was called instead of a.start(); so the entire program runs as a single thread, and a.run() is guaranteed to complete before the print statement is called.   |
| 3   | a Prints a number greater than or equal to | Thread a1 will run for at least ten  |

|   |        |  |  |
|---|--------|--|--|
|   | c      | 0 This program will run for at least ten seconds   | seconds, but the main method is likely to run to completion very quickly. The start method will return without waiting for thread a1 to complete. Since thread a1 immediately goes to sleep the thread that is processing the main method has an opportunity to complete the main method quickly. The number printed in the main method can be as small as zero.   |
| 4 | e      | volatile   | A field might be shared between two or more threads. Each thread is allowed to maintain a working copy of the field. If the threads do not reconcile the working copies then each might be working with a different value. The volatile modifier is used to force each thread to reconcile its working copy of the field with the master copy in main memory.  |
| 5 | a<br>c | The first number printed is greater than or equal to 0. The second number printed must always be greater than 5000 | The notify method is never invoked on thread a1; so it will sleep for at least five seconds. The invocation of the join method forces the main thread to wait for the completion of thread a1. The argument of 6000 will allow the main thread to wait for six seconds if necessary, but we know that thread a1 will complete in only five seconds. The first number printed will be greater than or equal to zero, and the second number will be greater than or equal to 5000. The synchronized block is necessary, because it is necessary to hold the lock of an object when the wait method is invoked. |
| 6 | d      | Prints: ABC  | The block inside the main method is synchronized on the String array object sa. Inside the block, a new thread t1 is started and will run at the discretion of the thread scheduler. The A.run method also contains a block that is synchronized on the String array object sa. Even if the thread scheduler moves thread t1 into the Running state, it will block while attempting to acquire the lock of the String array object sa. Thread t1 will continue to block until the synchronized block in the B.main method runs to  |

|    |                       |  |   |
|----|-----------------------|--|---|
|    |                       |  | completion. At that time, the contents of the String array object have all been updated.  |
| 7  | d Prints: ABC         |  | Inside the main method, thread t1 is started and will move into the Running state at the discretion of the thread scheduler. The A.run method invokes the wait method on the String array object sa causing the thread to block until another thread invokes the sa.notify method. Before the B.main method invokes sa.notify, all of the elements of the String array object sa have already been updated.   |
| 8  | b<br>c<br>d<br>e<br>h | The Thread.join method is always invoked on an instance of Thread. The Thread.join method causes the current thread to wait for the referenced thread to die. The Thread.join method declares an InterruptedException in the throws clause. The Thread.join method accepts a timeout value as an argument. A timeout of zero will allow Thread.join to wait forever if necessary | The Thread.join method is not static. Thread.join is always invoked on an instance of Thread. Thread.join causes the current thread to wait until the referenced thread has died. The maximum time limit to wait for the death of the referenced thread can be specified in milliseconds by an argument. Thread.join will throw an InterruptedException if the interrupt method is invoked on the current Thread.   |
| 9  | b<br>d                | By entering a synchronized instance method of the obj1 By entering the body of a block that is synchronized on obj1  | Blocking on I/O or invoking the Thread.sleep or Object.wait method causes a thread to enter the Not-Runnable state. Invoking the notify method on an object wakes up a thread that is waiting on the object. The thread that invokes wait or notify on an object should already hold the lock of the object. Invoking the wait or notify method does not cause the thread to hold the lock. Static methods are synchronized on the lock of the Class object of the class. Instance methods are synchronized on the lock of the instance of the class. |
| 10 | a<br>b<br>d<br>f      | Another thread invokes the notify method on the object, obj1, and T1 is selected to move out of the wait set Another thread invokes the notifyAll method on the object. Another thread interrupts thread T1. A specified timeout   |   |

|    |        |   |  |
|----|--------|---|--|
|    |        | period has elapsed  |  |
| 11 | e      | None of the above   | All of the class instance creation expressions are legal. The String instance A is the name of the Thread. Yes, the exam requires you to memorize the Thread constructor signatures.   |
| 12 | c      | 3   | The position of the arguments have been reversed in the constructor on line 3. The Runnable argument should appear before the thread name argument. Yes, the exam requires you to memorize the Thread constructor signatures.  |
| 13 | a      | The number printed is greater than or equal to 0                              | The main thread invokes the start method on thread a1. There is no way to predict when the new thread will start to run. At some point in time, the main thread will proceed to the print statement where the value of the startTime variable is subtracted from the current time. The value printed will be greater than or equal to one. At some point in time, thread a1 will begin to run and it will invoke the wait method. Since no other thread invokes the notify method on a1, it will wait forever, and the program will never run to completion. |
| 14 | a      | The number printed is greater than or equal to 0                              | The a1 thread is a daemon thread; so the program can run to completion even if thread a1 is still running, waiting or sleeping. The notify method is never invoked on thread a1. If thread a1 were not a daemon thread, then the program would wait forever. However, the program will run to completion without waiting for a1.   |
| 15 | b<br>c | Some or all of the numbers 0 through 9<br>could be printed Nothing is printed | All of the threads started in method B.m1 are daemon threads; so the program can run to completion even if some or all of the daemon threads have not run.   |
| 16 | e      | Compile-time error  | Remember that the Thread.start method is an instance method and can not be invoked from a static context.  |
| 17 | e      | Compile-time error  | Remember that the Object.notify method is an instance method and can not be invoked from a static context. Also, the thread that invokes the notify method on  |

|    |        |  |  |
|----|--------|--|--|
|    |        |  | an object must hold the lock of the object.  |
| 18 | d      | Compile-time error   | Both the sleep and join methods declare an InterruptedException that must be caught or declared in the throws clause of A.main.  |
| 19 | e      | An IllegalThreadStateException is thrown at run-time                           | For the purposes of the exam, invoking the start method on a thread that has already been started will generate an IllegalThreadStateException. The actual behavior of Java might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. However, for the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification of the Thread.start method and ignores the actual behavior of the 1.4 version of the JVM. The Thread.join method is included here to verify that the thread is already dead before the start method is invoked the second time. If this code is executed using the 1.4 version of the JVM, the exception will not be thrown. However, for the purposes of the exam, the exception is always thrown. The real exam question will probably not include the invocation of the join method. |
| 20 | b<br>d | Prints: [T1,A][T2,B] Prints:<br>[T2,B][T1,A]                                   | Since method m1 is synchronized, it is guaranteed that no more than one thread will execute the method at any one time. Even though the start method is invoked on thread T1 first, there is no guarantee that it will actually begin to run first.  |
| 21 | a      | The priority assigned to thread T2 is greater than the priority assigned to T1 | The Java Language Specification suggests that higher priority threads should be given preference over lower priority threads, but explicitly states that   |



the preference is not a guarantee. It is very important to remember that no guarantee exists.

---

---

## Corejava class programs

### Introduction programs

```
Class Car
{
    int wheels;
    int regno;
    int doors;

    void start()
    {
        System.out.println("the car is started ");
    }

    void stop()
    {
        System.out.println("the car is stopped");
    }

    public static void main(String args[])
    {
        Car c = new Car();
        c.start();
        c.stop();
    }
}
```

**or**

The above program can be rewriting by using the following way

```
class Car
{
    int wheels;
    int regno;
    int doors;

    void start()
    {
        System.out.println("the car is started ");
    }

    void stop()
    {
        System.out.println("the car is stopped");
    }
}

class cardemo
{
    public static void main(String args[])
    {
        Car c = new Car();
        c.start();
        c.stop();
    }
}
```

## Operators

```
public class TernaryOperatorsDemo {  
    public static void main(String args[]){  
        int x = 10, y = 12, z = 0;  
        z = x > y ? x : y;  
        System.out.println("z : "+z);  
    }  
}
```

```
public class RelationalOperatorsDemo  
{  
    public static void main(String args[])  
    {  
        int x = 10, y = 5;  
        System.out.println("x > y : "+(x > y));  
        System.out.println("x < y : "+(x < y));  
        System.out.println("x >= y : "+(x >= y));  
        System.out.println("x <= y : "+(x <= y));  
        System.out.println("x == y : "+(x == y));  
    }  
}
```

```
        System.out.println("x != y : "+(x != y));
    }

}
```

```
public class LogicalOperatorsDemo {
    public static void main(String args[]){
        boolean x = true;
        boolean y = false;
        System.out.println("x & y : "+(x & y));
        System.out.println("x && y : "+(x && y));
        System.out.println("x | y : "+(x | y));
        System.out.println("x || y: "+(x || y));
        System.out.println("x ^ y : "+(x ^ y));
        System.out.println("!x : "+(!x));
    }
}
```

```
public class CompoundOperatorsDemo {
    public static void main(String args[]){
        int x = 0, y = 5;
        x += 3;
```

```
System.out.println("x : "+x);
y *= x;
System.out.println("y : "+y);

/*Similarly other operators can be applied as shortcuts. Other
compound assignment operators include boolean logical
, bitwiseand shift operators*/
}
}
```

```
public class BitwiseOperatorsDemo {
    public static void main(String args[]){
        int x = 7;
        int y = 5;
        int z;
        System.out.println("x & y : "+(x & y));
        System.out.println("x | y : "+(x | y));
        System.out.println("x ^ y : "+(x ^ y));
        System.out.println("~x : "+(~x));
        System.out.println("x<< y : "+(x << y));
        System.out.println("x >> y : "+(x >> y));
    }
}
```

|                      |                               |
|----------------------|-------------------------------|
| postfix              | [] . () expr++ expr--         |
| unary                | ++expr --expr +expr -expr ! ~ |
| creation/caste       | new (type)expr                |
| multiplicative       | * / %                         |
| additive             | + -                           |
| shift                | << >> >>>                     |
| relational           | < <= > >= instanceof          |
| equality             | == !=                         |
| bitwise AND          | &                             |
| bitwise exclusive OR | ^                             |
| bitwise inclusive OR |                               |
| logical AND          | &&                            |
| logical OR           |                               |
| ternary              | ? :                           |
| assignment           | = "op="                       |

```

public class IfStatementDemo {

    public static void main(String[] args) {
        int a = 10, b = 20;
        if(a > b)
            System.out.println("a > b");
    }
}
```

```
    if(a < b)
        System.out.println("b > a");
}

}
```

```
public class IfElseStatementDemo {

    public static void main(String[] args) {
        int a = 10, b = 20;
        if(a > b){
            System.out.println("a > b");
        }else{
            System.out.println("b > a");
        }
    }
}
```

```
public class IfElseIfStatementDemo {

    public static void main(String[] args) {
        int a = 10, b = 20, c = 30;
        if(a > b && a > c){


```

```
        System.out.println("a is the greatest");
    }else if(b > c){
        System.out.println("b is the greatest");
    }else{
        System.out.println("c is the greatest");
    }
}
```

```
public class SwitchCaseStatementDemo {

    public static void main(String[] args) {
        int a = 10, b = 20, c = 30;
        int status = -1;
        if(a > b && a > c){
            status = 1;
        }else if(b > c){
            status = 2;
        }else{
            status = 3;
        }

        switch(status){
            case 1: System.out.println("a is the greatest");
                      break;
            case 2: System.out.println("b is the greatest");
                      break;
            case 3: System.out.println("c is the greatest");
                      break;
            default: System.out.println("Cannot be determined");
        }
    }
}
```

```
public class WhileLoopDemo {
```

```
public static void main(String[] args) {  
    int count = 1;  
    System.out.println("Printing Numbers from 1 to 10");  
    while( count <= 10){  
        System.out.println(count++);  
    }  
}
```

```
public class DoWhileLoopDemo {  
  
    public static void main(String[] args) {  
        int count = 1;  
        System.out.println("Printing Numbers from 1 to 10");  
        do{  
            System.out.println(count++);  
        }while( count <= 10);  
    }  
}
```

```
public class Fibonacci{  
    public static void main(String args[]){  
  
        System.out.println("Printing Limited set of Fibonacci Sequence");  
        double fib1 = 0;  
        double fib2 = 1;  
        double temp = 0;  
  
        System.out.println(fib1);  
        System.out.println(fib2);
```

```

        do{
            temp = fib1 + fib2;
            System.out.println(temp);
            fib1 = fib2;           //Replace 2nd with first number
            fib2 = temp;          //Replace temp number with 2nd
        number
        }while(fib2 < 5000);
    }
}

class Demo
{
public static void main(String args[])
{
    int i=0;
    for( ; i<=10;i++)
        System.out.println("i value is:"+i);
}
(or)
for(;i<=10;)
    System.out.println("i value is:" + i++);
(or)
for( ; )
    System.out.println(" I value is :" + i++);
    if(i>10)
        break;

```

```

public class TimesTableExample {
    public static void main( String[] args ) {
        for( int i=1; i<=10; i++ ) {

```

```
        for( int j=1; j<=10; j++ ) {
            System.out.print( (i*j) + "\t" );
        }
        System.out.println();
    }
}
```

```
public class TimesTableExample2 {
    public static void main( String[] args ) {
        outer:
        for( int i=1; i<=10; i++ ) {
            for( int j=1; j<=10; j++ ) {
                if( ( i*j ) == 25 ) {
                    break outer;
                }
                System.out.print( (i*j) + "\t" );
            }
            System.out.println();
        }
        System.out.println( "Done" );
    }
}
```

```
public class ContinueExample {
    public static void main(String[] args) {
```

```
System.out.println("Odd Numbers");
for (int i = 1; i <= 10; ++i) {
    if (i % 2 == 0) continue;
    // Rest of loop body skipped when i is even
    System.out.println(i + "\t");
}
}
```

```
class BreakExample
{
    public static void main(String[] args)
    {
        System.out.println("Numbers 1 - 10");
        for (int i = 1; ; ++i) {
            if (i == 11)
                break;
            // Rest of loop body skipped when i is even
            System.out.println(i + "\t");
        }
    }
}
```

return in main method will terminate the application

```
class demo
{
    public static void main(String args[])
}
```

```

{
    int x=1;
    System.out.println("before return");
    if(x==1)
        return; (or) system.exit(0);
    System.out.println("after return");
}

```

```

public class LabeledWhile
{

```

```

public static void main(String[] args)
{
    int i = 0;
    outer:
        while(true)
        {
            System.out.println("Outer while loop");
            while(true)
            {
                i++;
                System.out.println("i = " + i);
                if(i == 1)
                {
                    System.out.println("continue");
                    continue;
                }
                if(i == 3)
                {
                    System.out.println("continue outer");
                    continue outer;
                }
                if(i == 5)
                {
                    System.out.println("break");
                    break;
                }
                if(i == 7)
                {
                    System.out.println("break outer");
                    break outer;
                }
            }
        }
}
```

Labeled For:

```
public class LabeledFor {  
  
    public static void main(String[] args) {  
        int i = 0;  
        outer: // Can't have statements here  
        for(; true ;) { // infinite loop  
            inner: // Can't have statements here  
            for(; i < 10; i++) {  
                System.out.println("i = " + i);  
                if(i == 2) {  
                    System.out.println("continue");  
                    continue;  
                }  
                if(i == 3) {  
                    System.out.println("break");  
                    i++; // Otherwise i never  
                    // gets incremented.  
                    break;  
                }  
                if(i == 7) {  
                    System.out.println("continue outer");  
                    i++; // Otherwise i never  
                    // gets incremented.  
                    continue outer;  
                }  
                if(i == 8) {  
                    System.out.println("break outer");  
                    break outer;  
                }  
                for(int k = 0; k < 5; k++) {  
                    if(k == 3) {  
                        System.out.println("continue inner");  
                        continue inner;  
                    }  
                }  
            }  
        }  
    }  
}
```

```
}
```

```
}
```

```
}
```

```
}
```

## exercise programs

print numbers from 1 to 100

find the factorial of a given no

display the multiplication table

test the given number is even or odd

test the given number is prime or not

write a program to find the given no is palindrome or not

## Static members Test

```
public class StaticTest
{
    public static void main(String[] args)
    {
        // fill the staff array with three Employee objects
        Employee[] staff = new Employee[3];

        staff[0] = new Employee("Tom", 40000);
        staff[1] = new Employee("Dick", 60000);
        staff[2] = new Employee("Harry", 65000);

        // print out information about all Employee objects
        for (int i = 0; i < staff.length; i++)
        {
```

```

Employee e = staff[i];
e.setId();
System.out.println("name=" + e.getName()
+ ",id=" + e.getId()
+ ",salary=" + e.getSalary());
}

int n = Employee.getNextId(); // calls static method
System.out.println("Next available id=" + n);
}
}

class Employee
{
    public Employee(String n, double s)
    {
        name = n;
        salary = s;
        id = 0;
    }

    public String getName()
    {
        return name;
    }

    public double getSalary()
    {
        return salary;
    }

    public int getId()
    {
        return id;
    }

    public void setId()
    {
        id = nextId; // set id to next available id
        nextId++;
    }

    public static int getNextId()
    {
        return nextId; // returns static field
    }
}

```

```

public static void main(String[] args) // unit test
{
    Employee e = new Employee("Harry", 50000);
    System.out.println(e.getName() + " " + e.getSalary());
}

private String name;
private double salary;
private int id;
private static int nextId = 1;
}

```

The following program uses parameters to a method

```

class student
{
    int sno;
    int roomno;
    int marks;

    void setvalues()
    {
        int sno=101;
        roomno=1;
        marks=99;
    }

    void display()
    {
        System.out.println("the sno is :" +sno);
        System.out.println("the room no is:" +roomno);
        System.out.println("the marks are :" +marks);
    }
}

Class studentdemo
{
    Public static void main(String args[])
    {
        student s1 = new student();
        s1.setvalues();
        s1.display();
    }
}

```

```
public class SquareOverload {  
    public static int square( int n ) {  
        System.out.println( "Integer Square" );  
        return n*n;  
    }  
  
    public static long square( long l ) {  
        System.out.println( "Long Square" );  
        return l*l;  
    }  
  
    public static double square( double d ) {  
        System.out.println( "Double Square" );  
        return d*d;  
    }  
  
    public static void main( String[] args ) {  
        int n = 5;  
        long l = 100;  
        double d = 1000.0;  
        System.out.println( "n squared=" + square( n ) );  
        System.out.println( "l squared=" + square( l ) );  
        System.out.println( "d squared=" + square( d ) );  
    }  
}
```

```
public class Scope {  
    static int x = 5;  
  
    public static int timesX( int n ) {  
        int result = n*x;  
        return result;  
    }  
  
    public static void main( String[] args ) {  
        int m = 10;  
        System.out.println( "m times x = " + timesX( m ) );  
    }  
}
```

```
public class FactorialRecursive {  
    public static int factorial( int n ) {  
        if( n == 1 ) return 1;  
        return n * factorial( n-1 );  
    }  
  
    public static void main( String[] args ) {  
        System.out.println( "5 factorial = " + factorial( 5 ) );  
    }  
}
```

```
public class RecursiveIterativeTest {  
    public long factorialIterative( long n ) {  
        long result = n;  
        for( long i=n-1; i>0; i-- ) {  
            result *= i;  
        }  
        return result;  
    }  
  
    public long factorialRecursive( long n ) {  
        if( n == 1 ) return 1;  
        return n * factorialRecursive( n-1 );  
    }  
  
    public static void main( String[] args ) {  
        RecursiveIterativeTest test = new RecursiveIterativeTest();  
        long memBefore = Runtime.getRuntime().freeMemory();  
        long start = System.currentTimeMillis();  
        System.out.println( "Iterative factorial 1000 = " + test.factorialIterative( 20 ) );  
        long end = System.currentTimeMillis();  
        long memAfter = Runtime.getRuntime().freeMemory();  
        System.out.println( "Runtime: " + ( end - start ) + "ms, memory used=" + ( memAfter -  
memBefore ) );  
        memBefore = Runtime.getRuntime().freeMemory();  
        start = System.currentTimeMillis();  
        System.out.println( "Recursive factorial 1000 = " + test.factorialRecursive( 20 ) );  
        end = System.currentTimeMillis();  
        memAfter = Runtime.getRuntime().freeMemory();
```

```
        System.out.println( "Runtime: " + ( end - start ) + "ms, memory used=" + ( memAfter -  
memBefore ) );  
    }  
}
```

```
class arraydemo  
{  
    public static void main(String[] args)  
    {  
        int a[]={10,20,30,40,50};  
        for(int i=0;i<5;i++)  
        {  
            System.out.println(a[i]);  
        }  
    }  
}
```

```
class A  
{  
    public String toString()  
    {  
        return "satya technologies";  
    }  
}
```

```
public static void main(String[] args)
{
    A a1 = new A();
    A a2 = new A();
    System.out.println(a1);
    System.out.println(a2);

}

import java.util.*;

class ArrayListDemo
{
    public static void main(String[] args)
    {
        ArrayList a1 = new ArrayList(20);
        System.out.println("the size is "+a1.size());
        a1.add("10");
        a1.add("30");
        a1.add("40");
        a1.add("20");
        a1.add("20");
        a1.add(null);
        a1.add(null);
        a1.add(3,"50");

        System.out.println(a1);
    }
}
```

```
class arr_clone
{
    public static void main(String[] args)
    {
        int arr[]={11,22,33};
        int brr[]=(int[])arr.clone();
        arr[1]=100;
        for(int i=0;i<arr.length;i++)
            System.out.println(arr[i]);

        for(int i=0;i<brr.length;i++)
            System.out.println(brr[i]);
    }
}
```

```
class assertiontest
{
    public static void main(String[] args)
    {
```

```
int a=10;  
int b=0;  
  
assert a>0 : "value is less than zero";  
assert b>0 : "value is less than zero";  
  
int c=a/b;  
  
System.out.println("the c value is:"+c);  
}  
}
```

The following program is used for calling A class main method from B class

---

```
class A  
{  
    public static void main(String[] args)  
    {  
        System.out.println(args[0]);  
        System.out.println(args[1]);  
    }  
}  
class B  
{  
    public static void main(String[] args)  
    {  
        String arg[]={ "hai", "hello" };  
        A.main(arg);  
    }  
}
```

This program is for command line arguments

---

Compile

    Javac A.java

Execute

    Java A 10 20

```
class A
{
    public static void main(String[] args)
    {
        int x=Integer.parseInt(args[0]);
        int y=Integer.parseInt(args[1]);

        int z=x+y;
        System.out.println("the sum is :" +z);
    }
}
```

This program for creating the userdefined checked exceptions

---

Class InsufficientFundsException extends Exception  
{}

```
class Bank
{
    public static void main(String[] args) throws InsufficientFundsException
    {
        int bal=1000;
        int withdrawn=2000;
        if(bal<withdrawn)
            throw new InsufficientFundsException("bal is less than withdrawn");
        else
            System.out.println("proceed with transaction");
    }
}
```

This program is for creating the userdefined unchecked exception

---

```
Class InsufficientFundsException extends RuntimeException
{
    InsufficientFundsException(String s)
    {
        Super(s);
    }
}

class Bank
{
    public static void main(String[] args) throws InsufficientFundsException
    {
        int bal=1000;
        int withdrawn=2000;
        if(bal<withdrawn)
            throw new InsufficientFundsException("bal is less than withdrawn");
        else
            System.out.println("proceed with transaction");
    }
}
```

The following program displays the runtime polymorphism is not possible in case of static methods(static methods not participating in method overriding)

---

```
class AA
{
    static void classmethod()
    {
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}
class staticoverride
{
    public static void main(String args[])
    {
        AA a1 = new AA();
```

```
BB b1 = new BB();

AA x;
x=a1;
x.classmethod();
x.instancemethod();

x=b1;
x.classmethod();
x.instancemethod();
}

}
```

Program shows bytarrayinputstream

---

```
import java.io.*;

class bisdemo
{
    public static void main(String[] args) throws Exception
    {
        String s = "satya gives concept oriented java";
        byte b1[]={s.getBytes()};
        ByteArrayInputStream bis = new ByteArrayInputStream(b1);

        boolean b2 = bis.markSupported();
        System.out.println("mark supported! "+b2);

        bis.skip(5);

        int k;
        while((k=bis.read())!=-1)
        {
            System.out.print((char)k);
        }
    }
}
```

Constructor overloading and calling super class constructors using super keyword

---

```
class box
{
    double width;
    double height;
    double depth;
    box()
    {
        width=10.5;
        height=20.5;
        depth=30.5;
    }
    box(double x,double y,double z)
    {
        width=x;
        height=y;
        depth=z;
    }
    box(double x)
    {
        width=height=depth=x;
    }
    void output()
    {
        System.out.println("width is :" + width);
        System.out.println("height is :" + height);
        System.out.println("depth is :" + depth);
    }
}
```

```
class boxweight extends box
{
    double weight;
    boxweight()
    {
        super();
        weight=80.5;
    }
    boxweight(double p,double q,double r,double s)
    {
        super(p,q,r);
        weight=s;
    }
    boxweight(double x)
    {
        //width=height=depth=weight=x;
        super(x);
        weight=x;
    }
    void display()
    {
        System.out.println("width is :" + width);
        System.out.println("height is :" + height);
        System.out.println("depth is :" + depth);
        System.out.println("weight is :" + weight);
    }
}
class boxdemo
{
    public static void main(String[] args)
    {
        boxweight b1 = new boxweight();
        boxweight b2 = new boxweight(55.5,66.6,77.7,88.8);
        boxweight b3 = new boxweight(99.9);

        b1.display();
        b2.display();
        b3.display();
    }
}
```

## Collection frame work Calendar class

---

```
//CalendarTest.java
import java.util.*;
public class calendartest
{
    public static void main(String args[])
    {
        GregorianCalendar cal = new GregorianCalendar();
        System.out.println("Day is : " + cal.get(Calendar.DAY_OF_MONTH));
        System.out.println("Month is : " + cal.get(Calendar.MONTH));
        System.out.println("Year is : " + cal.get(Calendar.YEAR));
        if (cal.get(Calendar.ERA) == GregorianCalendar.AD)
            System.out.println("Era is : AD");
        else
            System.out.println("Era is : BC");
        if (cal.isLeapYear(cal.get(Calendar.YEAR)))
            System.out.println("Year is leap");
        else
            System.out.println("Year is non-leap");
    }
}
```

## Checkbox demo using awt

---

```
import java.awt.*;  
  
class CheckDemo extends Frame  
{  
    CheckDemo()  
    {  
        setLayout(new FlowLayout());  
  
        CheckboxGroup cbg = new CheckboxGroup();  
  
        Checkbox cb1 = new Checkbox("corejava",cbg,false);  
        Checkbox cb2 = new Checkbox("advjava",cbg,true);  
        Checkbox cb3 = new Checkbox("j2EE",cbg,false);  
        add(cb1);  
        add(cb2);  
        add(cb3);  
        setSize(300,400);  
        show();  
    }  
    public static void main(String[] args)  
    {  
        new CheckDemo();  
    }  
}
```

## Choice box using awt

---

```
import java.awt.*;  
  
class ChoiceDemo extends Frame  
{  
    ChoiceDemo()  
    {  
        setLayout(new FlowLayout());  
        Choice c = new Choice();  
        c.addItem("corejava");  
        c.addItem("adv.java");  
        c.addItem("j2ee");  
        c.addItem(".net");  
  
        add(c);  
        setSize(300,400);  
        show();  
    }  
    public static void main(String[] args)  
    {  
        new ChoiceDemo();  
    }  
}
```

```
import java.util.ArrayList;
import java.util.Iterator;

class Int {
    private int i;

    public Int(int ii) {
        i = ii;
    }

    public void increment() {
        i++;
    }

    public String toString() {
        return Integer.toString(i);
    }
}

public class Cloning {

    public static void main(String[] args) {
        ArrayList v = new ArrayList();
        for (int i = 0; i < 10; i++)
            v.add(new Int(i));
        System.out.println("v: " + v);
        ArrayList v2 = (ArrayList) v.clone();
        // Increment all v2's elements:
        for (Iterator e = v2.iterator(); e.hasNext();)
            ((Int) e.next()).increment();
        // See if it changed v's elements:
        System.out.println("v: " + v);
    }
}
```

```
}
```

Generating read-only collections or unmodifiable collections

---

```
import java.util.*;  
  
class collecreadonly  
{  
    public static void main(String args[])  
    {  
        ArrayList a1 = new ArrayList();  
        a1.add("a");  
        a1.add("b");  
        a1.add("x");  
        a1.add("y");  
        List li = new ArrayList(a1);  
        li=Collections.unmodifiableList(li);  
        try  
        {  
            li.set(0,"hello");  
        }  
        catch(UnsupportedOperationException e)  
        {  
            System.out.println(e);  
        }  
    }  
}
```

---

```
// -----  
  
Vector v1 = new Vector();  
v1.add("a");  
v1.add("b");  
v1.add("x");  
v1.add("y");  
  
List l1 = new Vector(v1);  
l1=Collections.unmodifiableList(l1);
```

```

        try
        {
            l1.set(0,"hello");
        }
        catch(UnsupportedOperationException e)
        {
            System.out.println(e);
        }
    // List stuff=Arrays.asList(new String[]{"a","b"});
}

```

User-defined sorting order using comparator interface

---

```

import java.util.*;
class Mycomp implements Comparator
{
    public int compare(Object x,Object y)
    {
        String a,b;
        a=(String)x;
        b=(String)y;
        int i,j,k;
        i=a.lastIndexOf(' ');
        j=b.lastIndexOf(' ');
        k=a.substring(i).compareTo(b.substring(j));
        if(k==0)
            return a.compareTo(b);
        else
            return k;
    }
}
class CompDemo1
{
public static void main(String[] args)
{
    TreeSet t = new TreeSet(new Mycomp());
    t.add("rami reddy");
    t.add("mohan kishore");
}

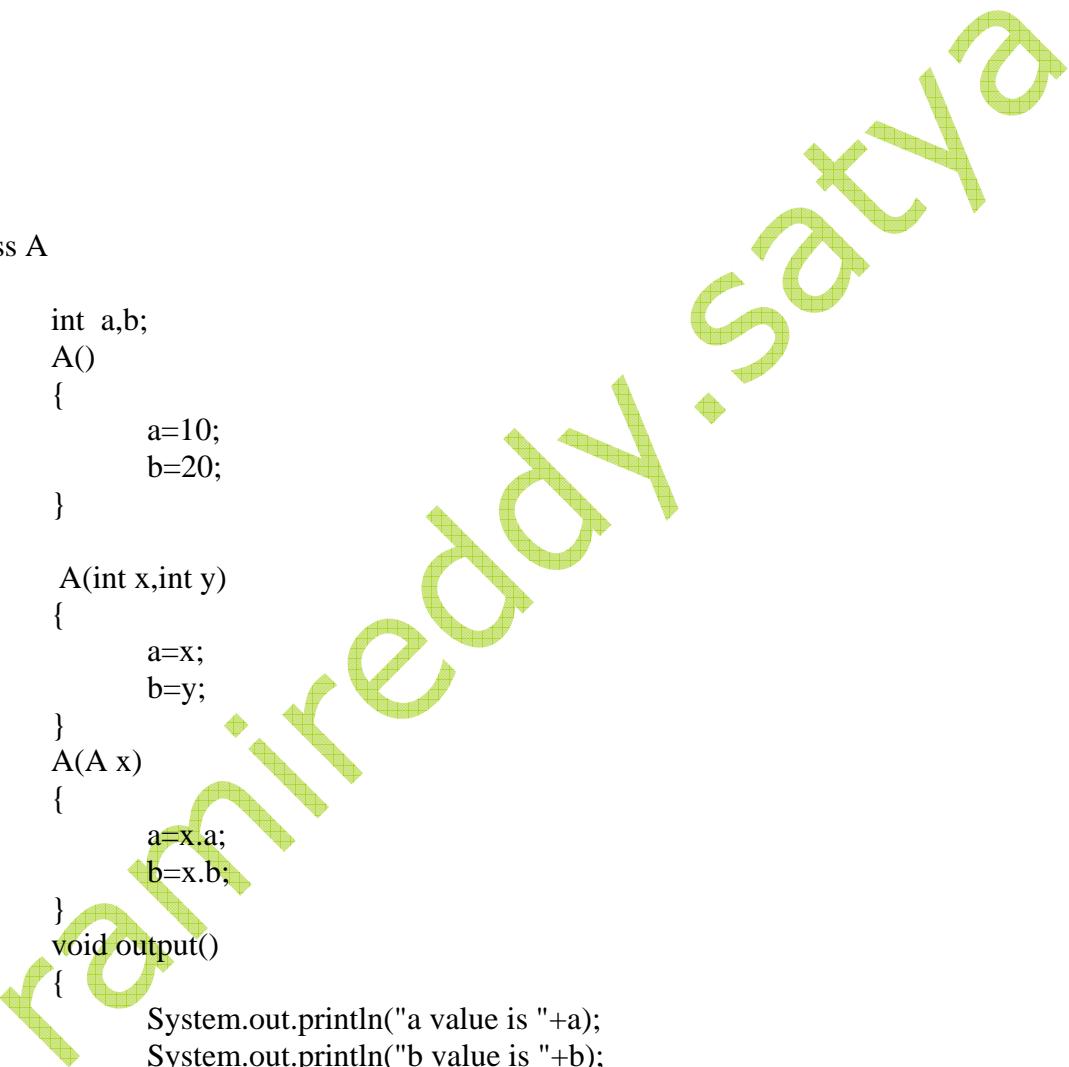
```

```
t.add("hari gupta");
t.add("nikhil sastry");

        System.out.println(t);
    }
}
```

Default Constructor is supplied by system or not(program gives compilation error)

```
-----  
class A  
{  
    int a,b;  
    A(int x,int y)  
    {  
        a=x;  
        b=y;  
    }  
}  
class Con1  
{  
    public static void main(String[] args)  
    {  
        A a1=new A();  
        A a2=new A(55,66);  
    }  
}
```



```
class A
{
    int a,b;
    A()
    {
        a=10;
        b=20;
    }

    A(int x,int y)
    {
        a=x;
        b=y;
    }

    A(A x)
    {
        a=x.a;
        b=x.b;
    }

    void output()
    {
        System.out.println("a value is "+a);
        System.out.println("b value is "+b);
    }
}

class ConstDemo
{
    public static void main(String[] args)
    {
        A a1 = new A();
        A a2 = new A(77,88);
    }
}
```

```

        A a3 =new A(a2);

        a1.output();
        a2.output();
        a3.output();

    }

}

class Test
{
    int a,b,c;
    Test(int x,int y)
    {
        a=x;
        b=y;
    }
    Test(int x,int y,int z)
    {
        this(x,y);
        c=z;
    }
    void display()
    {
        System.out.println("a value is :" +a+ " " +"b value is :" +b+ " "+"c value is
        :" +c);
    }
}
class constdemo1
{
    public static void main(String[] args)
    {
        Test t1 = new Test(22,33,44);
        t1.display();
    }
}

```

```
class assignarrayreference
{
    public static void main(String[] args)
    {
        int a[]={ 10,20,30,40};
        int b[] = new int[4];
        b=a;
        for(int i=0;i<b.length;i++)
        {
            System.out.println(b[i]);
        }
    }
}
```

## Dead-lock demo program

---

```
class A
{
    synchronized void show(B b)
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+"\tEntered A.show()");

        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            System.out.println("An Interruption has Raised");
        }
        System.out.println("Trying to call B.last()");
        System.out.println(Thread.currentThread().getName());
        b.last();
    }
    synchronized void last()
    {
        System.out.println("Inside A.Last");
    }
}
class B
{
    synchronized void display(A a)
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+"\tEntered B.display()");
```

```

try
{
    Thread.sleep(1000);
}
catch(InterruptedException e)
{
    System.out.println("An Exception has Raised");
}
System.out.println("Trying to call A.last()");
System.out.println(Thread.currentThread().getName());
a.last();
}
synchronized void last()
{
    System.out.println("Inside B.Last");
}
}
class Deadlock implements Runnable
{
    A a = new A();
    B b = new B();
    Deadlock()
    {
        Thread.currentThread().setName("Main Thread");
        Thread t=new Thread(this,"child Thread");
        //new Thread(this,"racing thread").start();

        t.start();
        a.show(b);
        System.out.println("Back in Main Thread");
    }
    public void run()
    {
        b.display(a);
        System.out.println("Back in Other Thread");
    }
    public static void main(String[] args)
    {
        new Deadlock();
    }
}/*

```

## Program for deep cloning and shallow cloning

---

The default/conventional behavior of `clone()` is to do a shallow copy. You can either override `clone()` to do a deep copy or provide a separate method to do a deep clone.

The simplest approach to deep cloning is to use Java serialization, where you serialize and deserialize the object and return the deserialized version. This will be a deep copy/clone, assuming everything in the tree is serializable. If everything is not serializable, you'll have to implement the deep cloning behavior yourself.

Assuming everything is serializable, the following should create a complete deep copy of the current class instance:

```
*/  
import java.io.*;  
  
class MyTest1 implements Serializable{  
    int dx,dy;  
  
    public MyTest1(int dx,int dy){  
        this.dx=dx;  
        this.dy=dy;  
    }  
    public void display(){  
        System.out.println("dx="+dx+" dy="+dy);  
    }  
}
```

```
}

public void modify(){
dx++;
dy++;
}

}

class deep{
public static void main(String args[]) throws Exception{
MyTest1 test=new MyTest1(100,200);
ByteArrayOutputStream baos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(baos);
oos.writeObject(test);
ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
ObjectInputStream ois = new ObjectInputStream(bais);
MyTest1 deepCopy = (MyTest1)ois.readObject();
test.display();
deepCopy.display();
test.modify();
System.out.println("after modification of original object ");
test.display();
deepCopy.display();
}
}
```

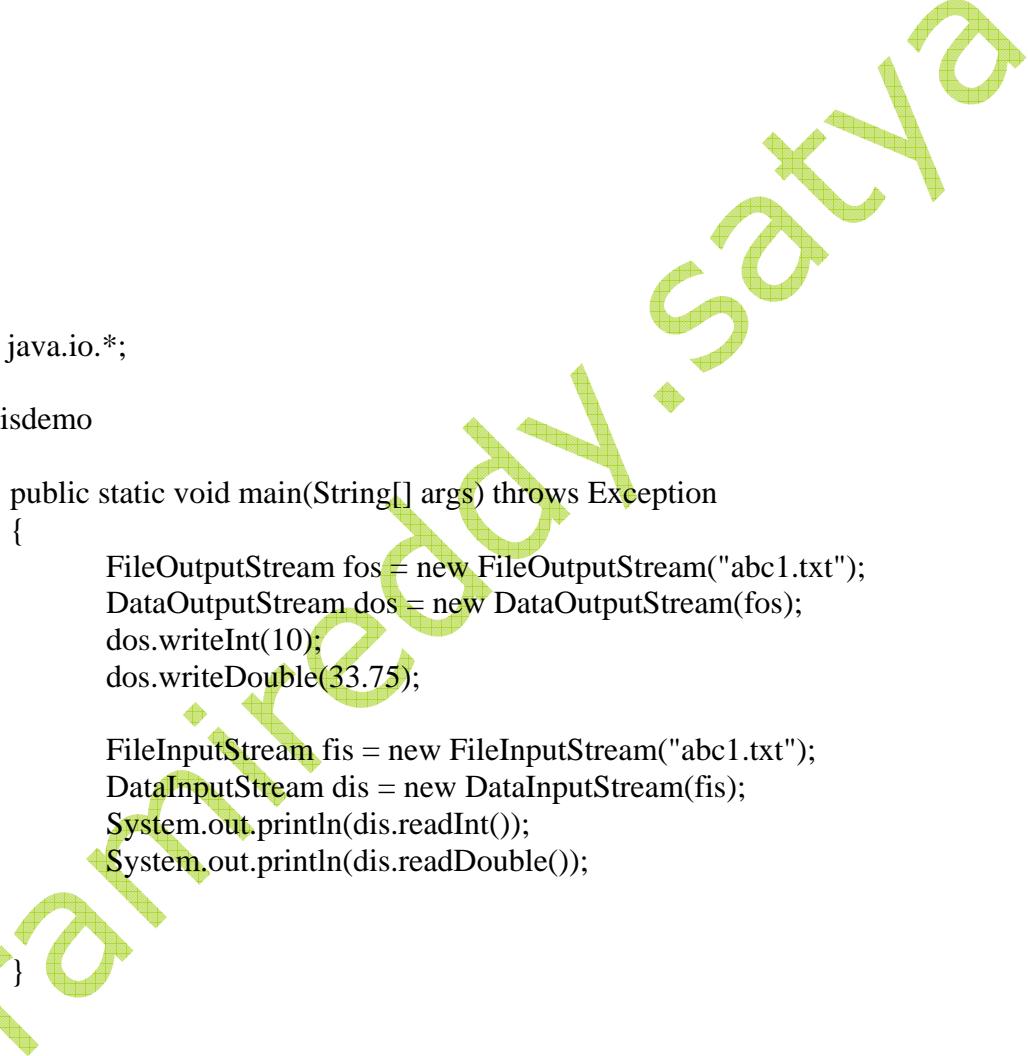
```
class A
{
    int x;
    A()
    {
        x=10;
    }
    void show()
    {
        System.out.println("x value is:"+x);
    }
}
class B extends A
{
    int y;
    B()
    {
        x=100;
        y=200;
    }
    void show()
    {
        System.out.println("y value is :" +y);
        super.show();
    }
}
class Demo
{
    public static void main(String[] args)
    {
        B b1 = new B();
        b1.show();
```

```
        }  
    }
```

Program to extract files from the directory

---

```
import java.io.*;  
  
class dirdemo  
{  
    public static void main(String[] args)  
    {  
        File f = new File("/work6pm");  
        String s[]=f.list(); //method retrieves files from a directory  
  
        for(int i=0;i<s.length;i++)  
        {  
            System.out.println(s[i]);  
        }  
    }  
}
```



```
import java.io.*;  
  
class disdemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileOutputStream fos = new FileOutputStream("abc1.txt");  
        DataOutputStream dos = new DataOutputStream(fos);  
        dos.writeInt(10);  
        dos.writeDouble(33.75);  
  
        FileInputStream fis = new FileInputStream("abc1.txt");  
        DataInputStream dis = new DataInputStream(fis);  
        System.out.println(dis.readInt());  
        System.out.println(dis.readDouble());  
    }  
}
```

## Program on deadlock

---

```
class DLDemo implements Runnable
{
    DLDemo x;
    public static void main(String[] args)
    {
        DLDemo dd1 = new DLDemo();
        DLDemo dd2 = new DLDemo();
        System.out.println(dd1);
        System.out.println(dd2);

        Thread t1 = new Thread(dd1,"first");
        Thread t2 = new Thread(dd2,"second");

        dd1.x=dd2;
        System.out.println(dd1.x);
        dd2.x=dd1;
        System.out.println(dd2.x);

        t1.start();
        t2.start();

        try
        {
            t1.join();
            t2.join();
        }
        catch(Exception e)
        {
            System.exit(0);
        }
    }
}
```

```
public synchronized void run()
{
    System.out.println(Thread.currentThread().getName());
    try
    {
        System.out.println("before sleep");
        Thread.sleep(1000);
        System.out.println("after
sleep"+Thread.currentThread().getName());
    }
    catch(InterruptedException e)
    {
        System.out.println(e);
    }
    System.out.println(Thread.currentThread().getName());
    System.out.println("before method");
    x.synmethod();
    System.out.println(Thread.currentThread().getName());
}
public void synmethod()
{
    System.out.println(Thread.currentThread().getName());
    try
    {
        Thread.sleep(1000);
    }
    catch(InterruptedException e)
    {
        System.out.println(e);
    }
    System.out.println("in synmethod");
}
```

### Program on single level inheritance

---

```
class Emp
{
    int eno;
    String ename;
    String eadd;
    void getEmp()
    {
        eno=101;
        ename="ram";
        eadd="hyd";
    }
    void empDisplay()
    {
        System.out.println("eno is :" + eno);
        System.out.println("ename is :" + ename);
        System.out.println("eadd is :" + eadd);
    }
}
class Pemp extends Emp
{
    String dept, desig;
    void getPemp()
    {
        dept="computer";
        desig="programmer";
    }
    void pempDisplay()
    {
        System.out.println("dept is :" + dept);
        System.out.println("desig is :" + desig);
    }
}
```

```
class EmpDemo
{
    public static void main(String[] args)
    {
        Pemp p1 = new Pemp();
        p1.getEmp();
        p1.getPemp();

        p1.empDisplay();
        p1.pempDisplay();
    }
}
```

Using awt, the collection framework program demonstration

```
-----  
import java.awt.*;  
import java.awt.event.*;  
import java.util.*;  
import java.io.*;  
  
class EntryForm extends Frame implements ActionListener  
{  
    String sno,sname,smail;  
    String tuple[];  
    ArrayList tuplecollector;  
    Button submit,next,showall,exit;  
    TextField phoneno,name,mailid;  
    Label I_phoneno,I_name,I_mailid;  
    public void intializeComponents()  
    {  
        name=new TextField(20);  
        phoneno=new TextField(10);  
        mailid=new TextField(25);  
        I_name=new Label("Person Name");  
        I_phoneno=new Label("Phone Number");  
        I_mailid=new Label("Email id");  
        submit=new Button("Submit");  
        next=new Button("Next");  
        showall=new Button("Show");  
        exit=new Button("Exit");  
    }  
    public void addComponents()  
    {  
        name.setBounds(220,50,100,20);  
    }  
}
```

```
        add(name);
        phoneno.setBounds(220,90,100,20);
        add(phoneno);
        mailid.setBounds(220,130,110,20);
        add(mailid);
        I_name.setBounds(120,50,100,25);
        add(I_name);
        I_phoneno.setBounds(120,90,100,25);
        add(I_phoneno);
        I_mailid.setBounds(120,130,110,25);
        submit.setBounds(20,170,100,25);
        add(submit);
        next.setBounds(140,170,100,25);
        add(next);
        showall.setBounds(260,170,100,25);
        add(showall);
        exit.setBounds(380,170,100,25);
        add(exit);
    }
    public void addComponentListeners()
    {
        submit.setActionCommand("Submit");
        submit.addActionListener(this);
        next.setActionCommand("Next");
        next.addActionListener(this);
        showall.setActionCommand("Show");
        showall.addActionListener(this);
        exit.setActionCommand("Exit");
        exit.addActionListener(this);
    }
    public void showForm()
    {
        tuple=new String[3];
        tuplecollector=new ArrayList();
        setLayout(null);
        setTitle("Mail id Entry Form");
        intializeComponents();
        addComponents();
        addComponentListeners();
        setSize(500,200);
        setVisible(true);
        setResizable(false);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getActionCommand().equals("Next"))
```

```

{
    tuple=new String[3];

    System.out.println(name.getText()+"/"+phoneno.getText()/"+"+mailid.getText());
    tuple[0]=new String(name.getText());
    tuple[1]=new String(phoneno.getText());
    tuple[2]=new String(mailid.getText());
    tuplecollector.add(tuple);
    name.setText("");
    phoneno.setText("");
    mailid.setText("");
}
else
{
    if(e.getActionCommand().equals("Submit"))
    {
        String s[]={};
        //following things Will be displayed on console
        for(int i=0;i<tuplecollector.size();i++)
        {
            s=(String[])tuplecollector.get(i);
            System.out.println(s);
            System.out.println(s[0]+"/"+s[1]+"/"+s[2]);
        }
    }
    else
    if(e.getActionCommand().equals("Show"))
    {
        //showing all records report
        System.out.println("Show");
        this.hide();
        new DisplayForm(tuplecollector,this);
    }
    else
    if(e.getActionCommand().equals("Exit"))
    {
        System.exit(0);
    }
}
}

class DisplayForm extends Frame implements ActionListener
{
    ArrayList tc;
    Font f;
    Button eform;
    EntryForm ef;
    public DisplayForm(ArrayList tc,EntryForm ef)

```

```

{
    this.tc=tc;
    this.ef=ef;
    f=new Font("Helvitica",Font.BOLD,10);
    eform=new Button("Show Entry Form");
    eform.setBounds(400,20,100,25);
    add(eform);
    eform.addActionListener(this);
    setLayout(null);
    setTitle("Records Display Form");
    setSize(500,300);
    setVisible(true);
}
public void paint(Graphics g)
{
    String s[]=new String[3];
    g.setFont(f);
    g.drawString("Name", 20,50);
    g.drawString("Phone Num",140,50);
    g.drawString("Mail Id",300,50);
    g.drawLine(20,60,400,60);
    for(int i=0,x=20,y=80;i<tc.size();i++,y+=30)

    {
        s=(String[])tc.get(i);
        g.drawString(s[0],x,y);
        g.drawString(s[1],x+120,y);
        g.drawString(s[2],x+250,y);
    }
}
public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().equals("ShowEF"))
    {
        ef.show();
        this.dispose();
    }
}
}

//the main class which invokes Frame
class MailidStorage
{
    public static void main(String[] args)

```

```
{  
    EntryForm ef=new EntryForm();  
    ef.showForm();  
}  
}
```

#### Finding file attributes

---

```
import java.io.*;  
import java.util.Date;  
  
class FileDemo  
{  
    public static void main(String[] args)  
    {  
        File f1 = new File("abc.txt");  
        boolean b=f1.exists();  
        System.out.println("file exists yes or no"+b);  
        System.out.println("the length of file :" +f1.length());  
        System.out.println("the name of file:" +f1.getName());  
        System.out.println("the parent of file :" +f1.getParent());  
        System.out.println("the path of file is :" +f1.getPath());  
        System.out.println("the absolute path is :" +f1.getAbsolutePath());  
        System.out.println("the modified date is :" +(new Date(f1.lastModified())));  
    }  
}
```

Reading data from a file and storing data into a file

---

```
import java.io.*;  
  
class FileRead  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileInputStream fis = new FileInputStream("abc.txt");  
        FileOutputStream fos = new FileOutputStream("xyz.txt");  
  
        int k;  
        while((k=fis.read())!=-1)  
        {  
            System.out.print((char)k);  
            fos.write(k); //writing data onto disk  
        }  
        fos.close();  
        fis.close();  
    }  
}
```

```
import java.io.*;
class finallydemo
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis=null;

        try
        {
            fis = new FileInputStream("abc.txt");
            int a=10;
            int b=0;
            int c=a/b;
        }
        //catch(Exception e)
        //{
        //System.out.println(e);
        //}
        finally
        {
            System.out.println("we are in finally");

            //try
            //{
            fis.close();
            //}
            //catch(Exception e)
            //{
            //System.out.println(e);
        }
    }
}
```

```
//}

        }

    }

}

import java.io.*;
class finallytest
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis=null;
        try
        {
            fis= new FileInputStream("abc.txt");
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is :" +c);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        finally
        {
            System.out.println("we are in finally block");
            fis.close();
        }
    }
}
```

Creating thread using extends Thread class

```
-----  
class First extends Thread  
{  
    public void run()  
    {  
        for(int i=0;i<10;i++)  
        {  
            System.out.println(i);  
        }  
    }  
    public static void main(String[] args)  
    {  
        First f1 = new First();  
        First f2 = new First();  
  
        f1.start();  
        f2.start();  
    }  
}
```

Different types of exception messages displaying

---

```
class firstexp
{
    public static void main(String[] args)
    {
        try
        {
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is"+c);
            int d[]={10,20};
            d[5]=50;
        }
        //System.out.println("try is over and enter into catch block");
        catch(Exception e)
        {
            System.out.println(e);
            e.printStackTrace();
            System.out.println(e.getMessage());
            String s=e.toString();
            System.out.println(s);
        }
    }
}
```

```
//System.out.println("one");
//System.out.println("two");
//System.out.println("three");
//System.out.println("four");

    }
}
```

```
class FirstThread extends Thread
{
    public static void main(String[] args)
    {
        FirstThread ft = new FirstThread();
        ft.start();
        FirstThread ft1 = new FirstThread();
        ft1.start();
    }
    public void run()
    {
        System.out.println("control entered into run method");

        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" +i);
        }
    }
}
```

```
class Fourth extends Thread
{
    public static void main(String[] args)
    {
        Fourth f1 = new Fourth();
        System.out.println(f1);

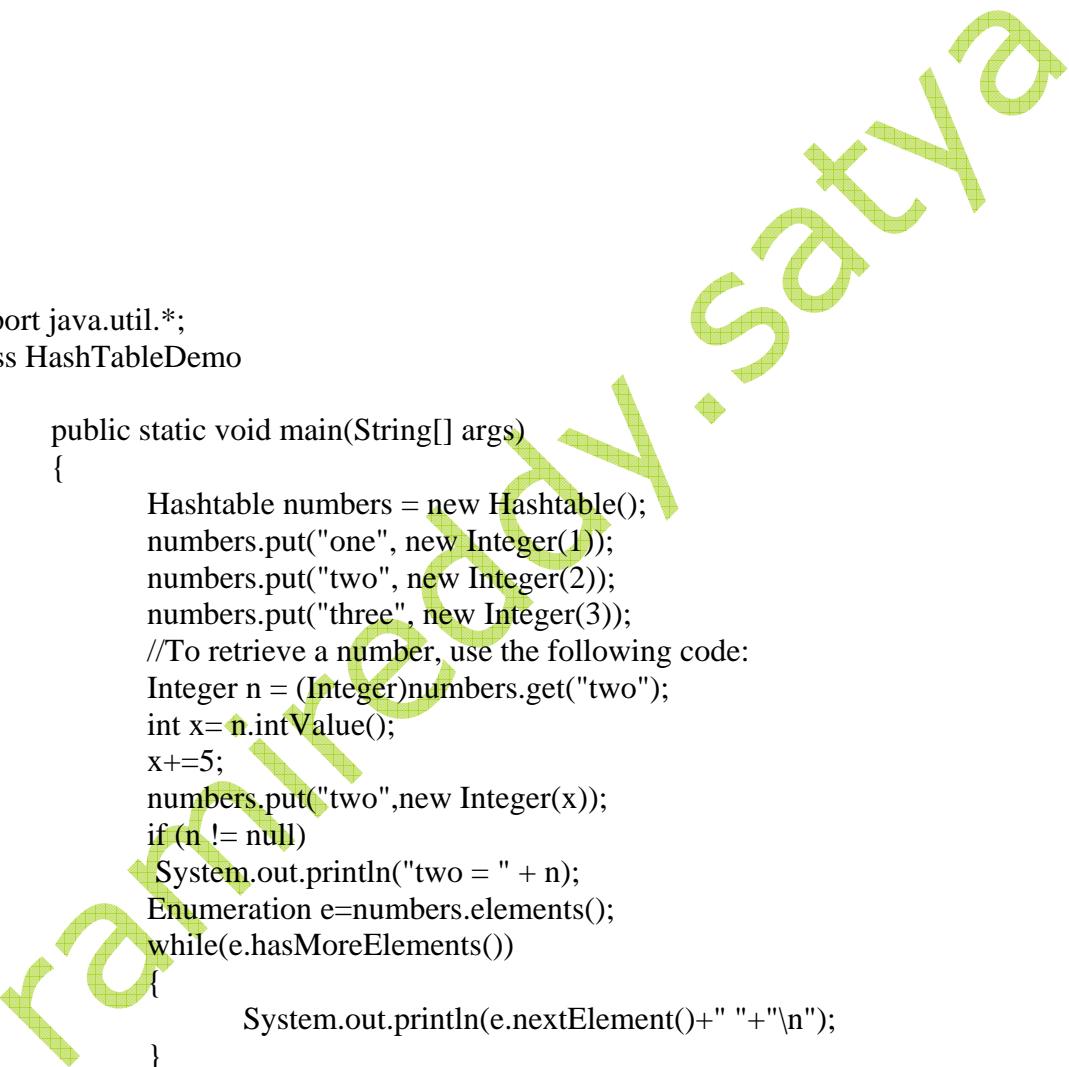
        Fourth f2 = new Fourth();
        System.out.println(f2);
    }
}
```

```
import java.util.*;

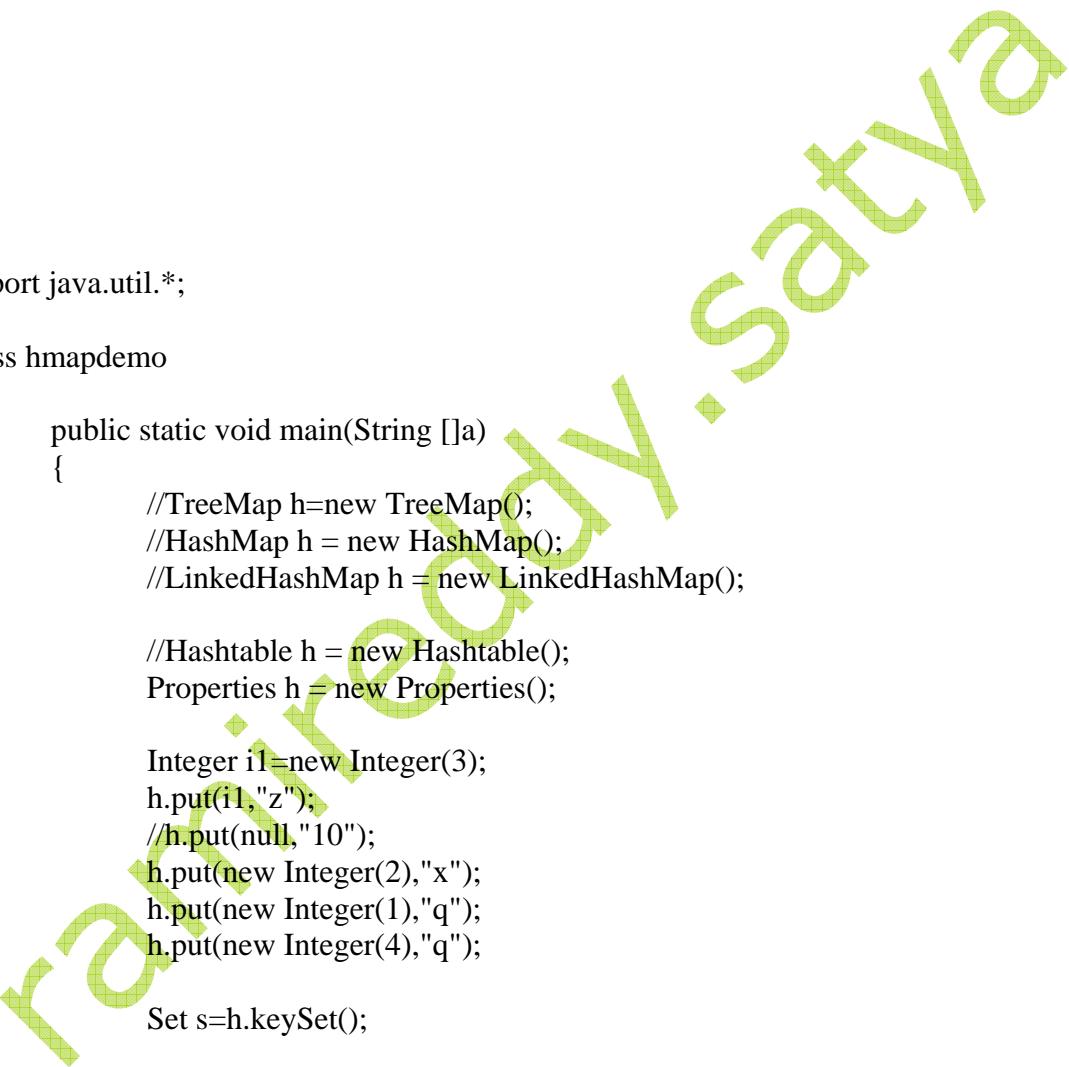
class hashsetdemo
{
    public static void main(String[] args)
    {
        HashSet h = new HashSet();

        h.add("10");
        h.add("20");
        h.add("30");
        h.add(null);
        h.add(null);
        h.add("30");

        System.out.println(h);
    }
}
```



```
import java.util.*;
class HashTableDemo
{
    public static void main(String[] args)
    {
        Hashtable numbers = new Hashtable();
        numbers.put("one", new Integer(1));
        numbers.put("two", new Integer(2));
        numbers.put("three", new Integer(3));
        //To retrieve a number, use the following code:
        Integer n = (Integer)numbers.get("two");
        int x=n.intValue();
        x+=5;
        numbers.put("two",new Integer(x));
        if(n != null)
            System.out.println("two = " + n);
        Enumeration e=numbers.elements();
        while(e.hasMoreElements())
        {
            System.out.println(e.nextElement()+" "+ "\n");
        }
    }
}
```



```
import java.util.*;

class hmapdemo
{
    public static void main(String []a)
    {
        //TreeMap h=new TreeMap();
        //HashMap h = new HashMap();
        //LinkedHashMap h = new LinkedHashMap();

        //Hashtable h = new Hashtable();
        Properties h = new Properties();

        Integer i1=new Integer(3);
        h.put(i1,"z");
        //h.put(null,"10");
        h.put(new Integer(2),"x");
        h.put(new Integer(1),"q");
        h.put(new Integer(4),"q");

        Set s=h.keySet();

        Iterator i=s.iterator();

        while(i.hasNext())
        {
            Object o=i.next();

            System.out.println("key="+o+" value="+h.get(o));
        }
    }
}
```

```
    }  
}
```

```
import java.util.*;  
class htabledemo  
{  
    public static void main(String a[])  
    {  
        Hashtable ht=new Hashtable();  
        ht.put(null,"activenet");  
        ht.put("a",new String("java2"));  
        ht.put("b",new String("c/c++"));  
        ht.put( "c",new String("java"));  
        ht.put("d",new String("java1"));  
  
        Set s=ht.keySet();  
        Object o1[]=s.toArray();  
        for(int i=0;i<=o1.length;i++)  
        {  
            System.out.println("the value is"+o1[i].toString());  
        }  
        Iterator i=s.iterator();  
        while(i.hasNext())  
        {  
            Object o=i.next();  
            System.out.println(ht.get(o));  
        }  
    }  
}
```

## Internationalized(i18n) program

---

```
import java.util.*;

public class i18nsample
{
    public static void main(String args[])
    {
        String language;
        String country;
        if(args.length!=2)
        {
            language=new String("en");
            country = new String("US");
        }
        else
        {
            language = new String(args[0]);
            country = new String(args[1]);
        }
        Locale currentlocale;
        ResourceBundle messages;

        currentlocale = new Locale(language,country);

        messages = ResourceBundle.getBundle("messagesbundle",currentlocale);

        System.out.println(messages.getString("greetings"));
        System.out.println(messages.getString("inquiry"));
        System.out.println(messages.getString("farewell"));
    }
}
```

```
import java.io.*;  
  
class inputdemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileInputStream fis = new FileInputStream("abc.txt");  
        int x=fis.available();  
  
        System.out.println("the available bytes are"+x);  
        boolean b = fis.markSupported();  
        System.out.println("the mark is supported or not"+b);  
  
        String s="hello how r u";  
        byte by[] = s.getBytes();  
        ByteArrayInputStream bis = new ByteArrayInputStream(by);  
  
        boolean b1 = bis.markSupported();  
        System.out.println(b1);  
    }  
}
```

```
interface calculate
{
    void sum();
    void sub();
    void mul();
}

class number implements calculate
{
    int a,b;
    void getdata()
    {
        a=10;
        b=20;
    }
    public void sum()
    {
        int x=a+b;
        System.out.println("the sum is :" +x);
    }
    public void sub()
    {
        int x=a-b;
        System.out.println("the sub is :" +x);
    }
    public void mul()
    {
        int x=a*b;
        System.out.println("the mul is :" +x);
    }
}
class interfacedemo
{
```

```

public static void main(String[] args)
{
    number n = new number();
    n.getdata();
    n.sum();
    n.sub();
    n.mul();
}
}

interface A
{
    void one();
}

interface B
{
    void two();
}

class Demo implements A,B
{
    public void one()
    {
        System.out.println("one method is implemented from A");
    }
    public void two()
    {
        System.out.println("two method is implemented from B");
    }
}
class interfacedemo0
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();
        d.two();

        A x;
        x=d;
        x.one();
    }
}

```

```
        x.two(); // this reference is not available  
                // in interface A  
    }  
}
```

```
interface A  
{  
    void one();  
}  
  
interface B  
{  
    void one();  
}  
  
class Demo implements A,B  
{  
    public void one()  
    {  
        System.out.println("one method is implemented");  
    }  
}  
class interfacedemo1  
{  
    public static void main(String[] args)  
    {  
        Demo d = new Demo();  
        d.one();  
  
        A x;  
        x=d;  
        x.one();  
  
        B x1;  
        x1=d;  
        x1.one();  
    }  
}
```

```
}
```

```
// program on same method with different return types
interface A
{
    int one();
}
interface B
{
    void one();
}

class Demo implements A,B
{
    public void one()
    {
        System.out.println("one method is implemented");
    }
    public int one()
    {
        System.out.println("one method with int return is implemented");
    }
}
class interfacedemo2
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();

        A x;
        x=d;
        x.one();

        B x1;
        x1=d;
    }
}
```

```
        x1.one();
    }
}
```

```
interface A
{
    void one();
}
abstract class B
{
    void one()
    {
        System.out.println("abstract class one method implemented");
    }
}

class Demo extends B implements A
{
    public void one()
    {
        System.out.println("one method is implemented in demo class");
    }
}

class interfacedemo3
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();
    }
}
```

```
interface A
{
    void one();
}

abstract class B implements A
{
    public void one()
    {
        System.out.println("abstract class one method implemented");
    }
}

class Demo extends B
{
    public void one()
    {
        System.out.println("one method is implemented in demo class");
    }
}

class interfacedemo4
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();

    }
}
```

## Program on ioredirection

---

```
import java.io.*;  
  
class ioredirect  
{  
    public static void main(String[] args)  
    {  
        FileInputStream fis = null;  
        try  
        {  
            fis=new FileInputStream(args[0]);  
        }  
        catch(FileNotFoundException e)  
        {  
            System.exit(0);  
        }  
        try  
        {  
            System.setIn(fis);  
            int i;  
            while((i=System.in.read())!=-1)  
                System.out.print((char)i);  
            fis.close();  
        }  
        catch(Exception e)  
        { }  
    }  
}
```

```
}
```

```
import java.io.*;  
  
class ioredirect2  
{  
public static void main(String[] args) throws Exception  
{  
    FileOutputStream fos = new FileOutputStream("ram.txt");  
    PrintStream pis = new PrintStream(fos);  
    System.setErr(pis);  
    int a=10;  
    int b=0;  
    int c=a/b;  
    //System.setOut(pis);  
    //System.out.print("hai hello how r u");  
}  
}
```

## Static objects

---

```
public class Iyear
{
    String classname,staffname;
    int no;
    Student stu[];
    public Iyear(String c,String s,int n,Student st[])
    {
        classname =c;
        staffname =s;
        no=n;
        stu = new Student[no];
        for(int i=0;i<no;i++)
            stu[i]=st[i];
    }
    public static void main(String[] args)
    {
        Iyear m;
        int mk1[]={73,84,95};
        int mk2[]={56,76,84};
        int mk3[]={73,84,95};
        int mk4[]={84,73,95};
        int mk5[]={95,73,84};

        Student st[];
        st = new Student[5];
        st[0]=new Student(2124,"hari",mk1);
```

```

        st[1]=new Student(2125,"mohan",mk2);
        st[2]=new Student(2126,"srinu",mk3);
        st[3]=new Student(2123,"prasanth",mk4);
        st[4]=new Student(2127,"ram",mk5);

        m= new Iyear("M.C.A","surya",5,st);
    }
}

```

```

class joindemo extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" +i);
        }
    }
    public static void main(String[] args) throws Exception
    {
        joindemo j1 = new joindemo();
        joindemo j2 = new joindemo();
        //try
        {
            j1.start();
            j1.join();

            j2.start();
            j2.join();
        //}
        //catch(Exception e)
        //{
        }
    }
}

```

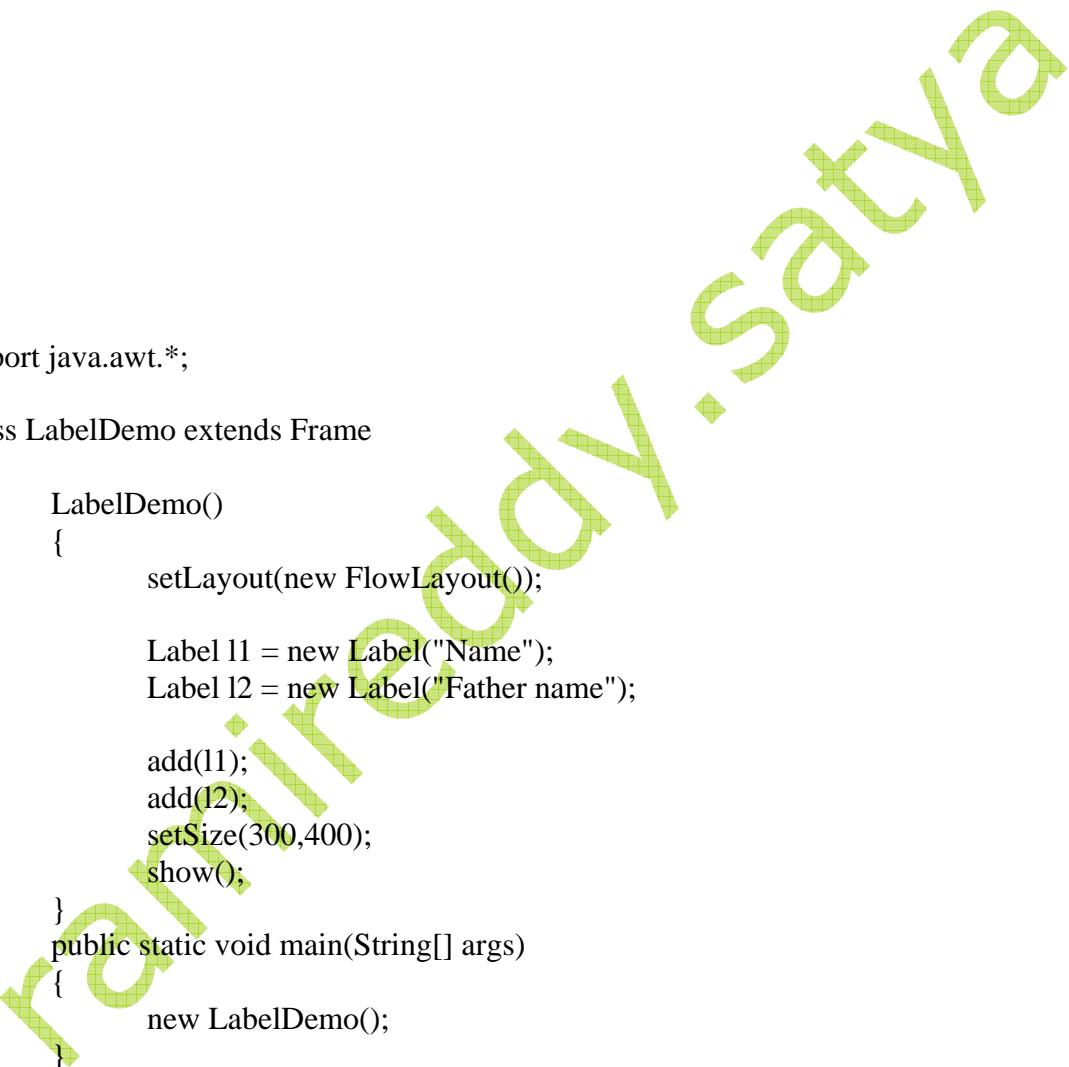
```
        System.out.println("the main method is completed");

    }

}
```

```
import java.io.*;

class KeyboardRead
{
    public static void main(String[] args) throws Exception
    {
        //FileInputStream fis = new FileInputStream("abc.txt");
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String s=null;
        System.out.println("enter text, if wants to close enter stop");
        do
        {
            s=br.readLine();
            System.out.println(s);
        }while(!s.equals("stop"));
    }//main
}//class
```



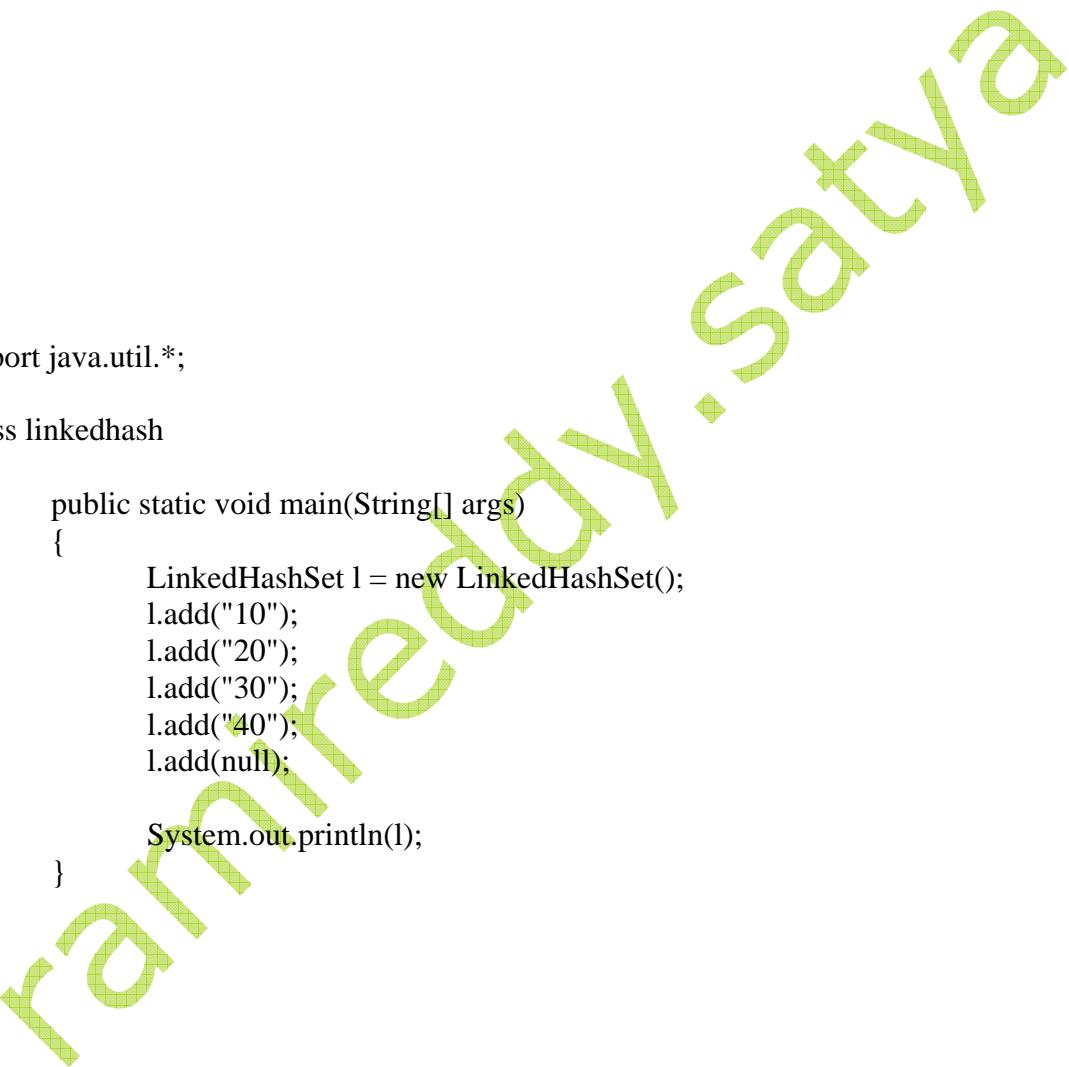
```
import java.awt.*;

class LabelDemo extends Frame
{
    LabelDemo()
    {
        setLayout(new FlowLayout());
        Label l1 = new Label("Name");
        Label l2 = new Label("Father name");
        add(l1);
        add(l2);
        setSize(300,400);
        show();
    }
    public static void main(String[] args)
    {
        new LabelDemo();
    }
}
```



```
import java.io.*;

class LineNumberDemo
{
    public static void main(String[] args)
    {
        try
        {
            FileInputStream fis = new FileInputStream(args[0]);
            LineNumberInputStream lis = new LineNumberInputStream(fis);
            DataInputStream dis = new DataInputStream(lis);
            String s=null;
            do
            {
                s = dis.readLine();
                System.out.println(lis.getLineNumber()+" "+s);
            }while(!s.equals(null));
        }
        catch(Exception e)
        {
        }
    }
}
```



```
import java.util.*;  
  
class linkedhash  
{  
    public static void main(String[] args)  
    {  
        LinkedHashSet l = new LinkedHashSet();  
        l.add("10");  
        l.add("20");  
        l.add("30");  
        l.add("40");  
        l.add(null);  
  
        System.out.println(l);  
    }  
}
```

```
import java.util.*;  
  
class LinkedListDemo  
{  
    public static void main(String[] args)  
    {  
        LinkedList l = new LinkedList();  
        l.add("10");  
        l.add("50");  
        l.add(null);  
        l.add(null);  
        l.add("50");  
        l.add("30");  
        l.add("20");  
        System.out.println(l);  
    }  
}  
  
import java.awt.*;  
  
class ListDemo extends Frame  
{  
    ListDemo()  
    {  
        setLayout(new FlowLayout());  
        List l = new List();  
        l.add("corejava");  
        l.add("adv.java");  
    }  
}
```

```
        l.add("j2ee");
        l.add(".net");

        add(l);
        setSize(300,400);
        show();
    }
    public static void main(String[] args)
    {
        new ListDemo();
    }
}
```

```
public class MarksOutOfBoundsException extends Exception
{
    MarksOutOfBoundsException(String s)
    {
        super(s);
    }
}
```

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="ApltoApl" width=400 height=300 name=a1>
<param name=p1 value=a2>
```

```

</applet>

<applet code="ApltoApl" width=400 height=300 name=a2>
<param name=p1 value=a1>
</applet>

*/



public class ApltoApl extends Applet implements ActionListener
{
    Button b1;
    String str;
    AppletContext ac;
    ApltoApl a;
    public void init()
    {
        b1 = new Button("red");
        b1.addActionListener(this);
        add(b1);
    }
    public void start()
    {
        ac=getAppletContext();
        a=(ApltoApl)ac.getApplet(getParameter("p1"));
    }
    public void actionPerformed(ActionEvent ae)
    {
        String s = ae.getActionCommand();
        if(s.equals("red"))
            a.setBackground(Color.red);
    }
}

```

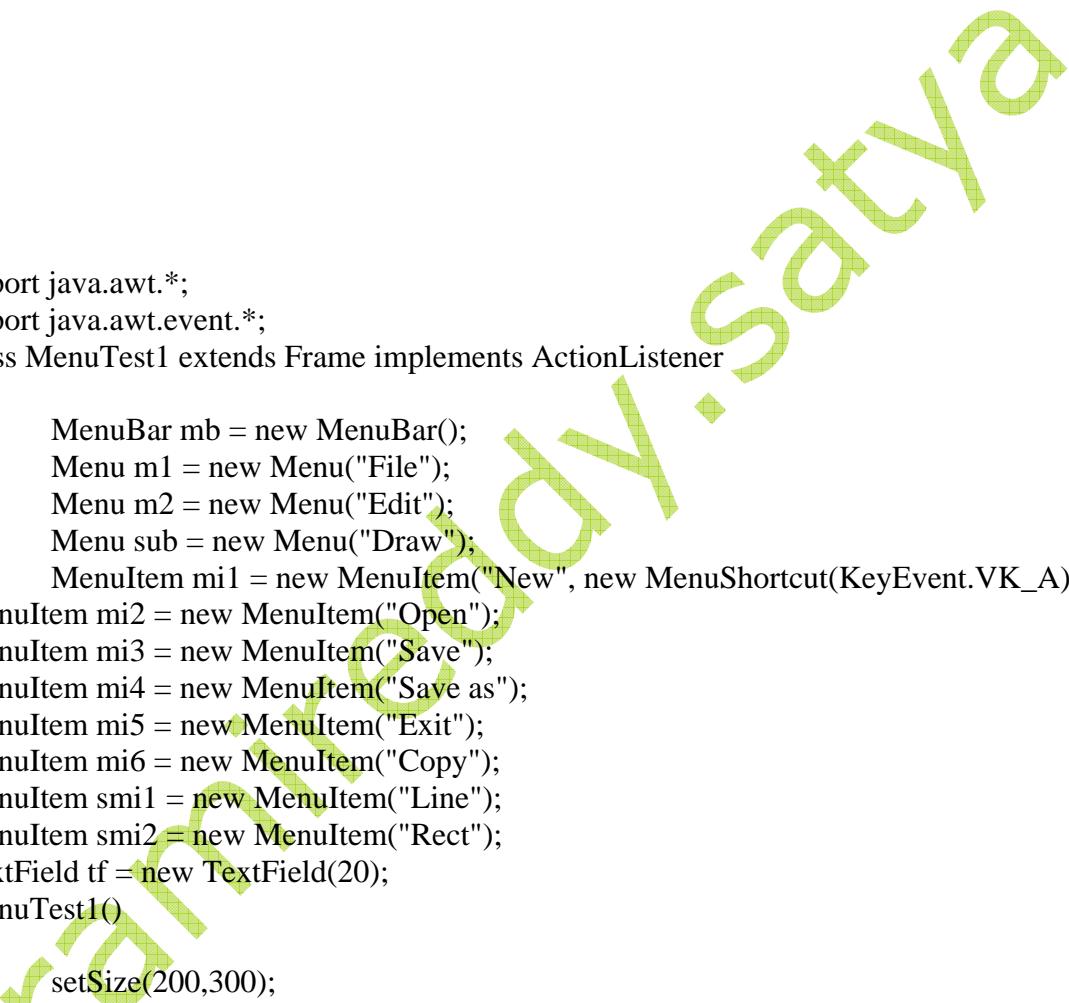
applets are basically used for developing the dynamic web pages like DHTML

applets are always accessed through internet browsers.(java enabled browsers)

applets are compiled java programs.

applets are java programs which can be embedded with html.

once the applet is loaded into client system it has no connections with server



```
import java.awt.*;
import java.awt.event.*;
class MenuTest1 extends Frame implements ActionListener
{
    MenuBar mb = new MenuBar();
    Menu m1 = new Menu("File");
    Menu m2 = new Menu("Edit");
    Menu sub = new Menu("Draw");
    MenuItem mi1 = new MenuItem("New", new MenuShortcut(KeyEvent.VK_A));
    MenuItem mi2 = new MenuItem("Open");
    MenuItem mi3 = new MenuItem("Save");
    MenuItem mi4 = new MenuItem("Save as");
    MenuItem mi5 = new MenuItem("Exit");
    MenuItem mi6 = new MenuItem("Copy");
    MenuItem smi1 = new MenuItem("Line");
    MenuItem smi2 = new MenuItem("Rect");
    TextField tf = new TextField(20);
    MenuTest1()
    {
        setSize(200,300);
        m1.add(mi1);
        m1.add(mi2);
        m1.add(mi3);
        m1.add(mi4);
        m1.addSeparator();
        m1.add(mi5);
        m1.add(mi6);
        m2.add(sub);
        sub.add(smi1);
        sub.add(smi2);
    }
}
```

```
mb.add(m1);
mb.add(m2);
setMenuBar(mb);
add(tf,"North");
mi1.addActionListener(this);
mi2.addActionListener(this);
mi3.addActionListener(this);
mi4.addActionListener(this);
mi5.addActionListener(this);
mi6.addActionListener(this);
smi1.addActionListener(this);
smi2.addActionListener(this);
setVisible(true);
}
public static void main(String args[])
{
new MenuTest1();
}
public void actionPerformed(ActionEvent ae)
{
    tf.setText("u selected"+ae.getActionCommand());
}
}
```

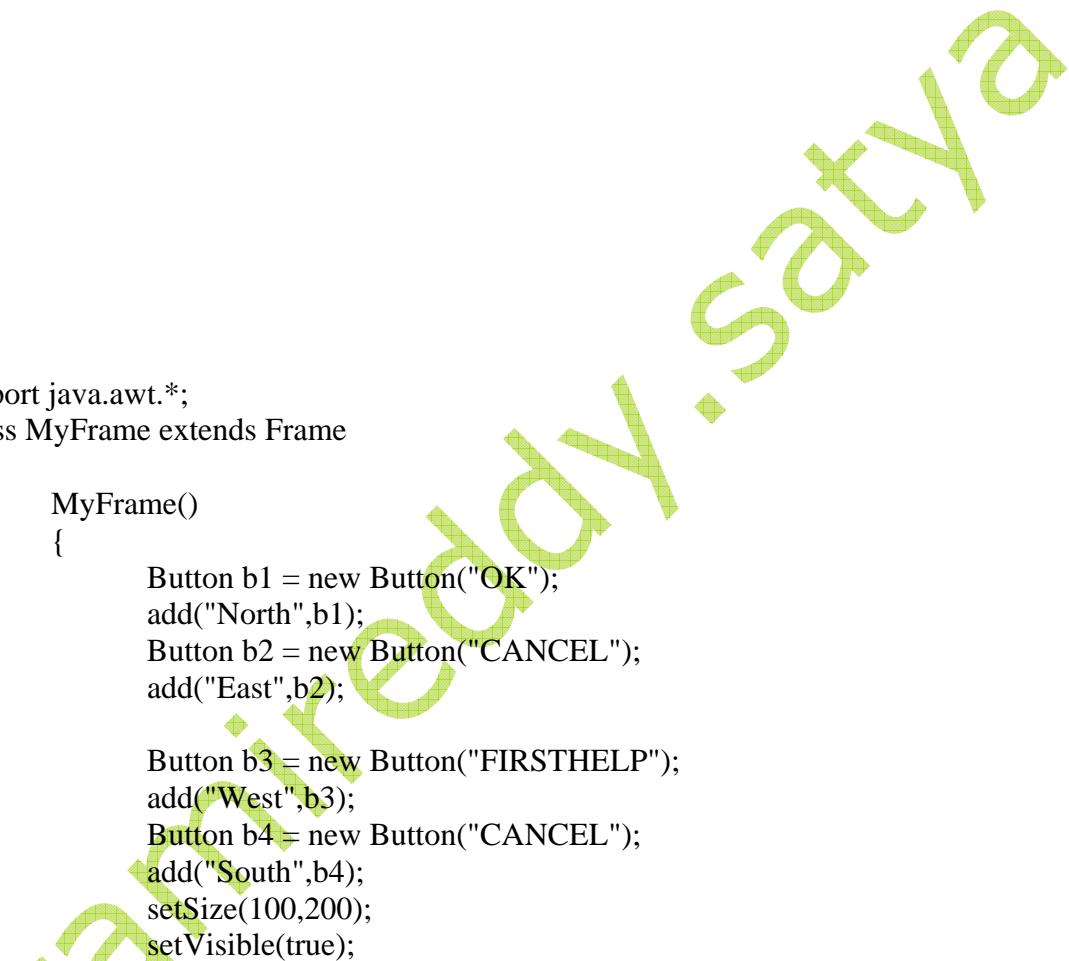
ramireddy.Satya

```
import java.awt.*;

class MyButton extends Frame
{
    MyButton()
    {
        Button b1 = new Button("OK");
    }
}
```

```
        add(b1);
        Button b2 = new Button("CANCEL");
        add(b2);
        setSize(300,400);
        setVisible(true);
    }
    public static void main(String[] args)
    {
        new MyButton();
    }
}
```

```
class mycomparable implements Comparable
{
    int m1,m2;
    public mycomparable(int x,int y)
    {
        m1=x;
        m2=y;
    }
    public int compareTo(Object obj)
    {
        mycomparable t = (mycomparable)obj;
        if((this.m1==t.m1)&&(this.m2==t.m2))
            return 1;
        else
            return 0;
    }
    public static void main(String args[])
    {
        mycomparable t1 = new mycomparable(10,20);
        mycomparable t2 = new mycomparable(10,20);
        if(t1.compareTo(t2)==1)
            System.out.println("both are same");
        else
            System.out.println("both are different");
    }
}
```



```
import java.awt.*;
class MyFrame extends Frame
{
    MyFrame()
    {
        Button b1 = new Button("OK");
        add("North",b1);
        Button b2 = new Button("CANCEL");
        add("East",b2);

        Button b3 = new Button("FIRSTHELP");
        add("West",b3);
        Button b4= new Button("CANCEL");
        add("South",b4);
        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
        new MyFrame();
    }
}
```

```
import java.awt.*;
class MyFrame1 extends Frame
{
    MyFrame1()
    {
        setLayout(new FlowLayout());

        Button b1 = new Button("OK");
        Button b2 = new Button("CANCEL");
        Button b3 = new Button("FIRSTHELP");
        Button b4 = new Button("CANCEL");
        Button b5 = new Button("FIRSTHELP");
        Button b6 = new Button("CANCEL");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);

        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
        new MyFrame1();
    }
}
```

```
}
```

```
import java.awt.*;
class MyFrame2 extends Frame
{
    MyFrame2()
    {
        setLayout(new GridLayout(3,4));

        Button b1 = new Button("OK");
        Button b2 = new Button("CANCEL");
        Button b3 = new Button("FIRSTHELP");
        Button b4 = new Button("CANCEL");
        Button b5 = new Button("FIRSTHELP");
        Button b6 = new Button("CANCEL");
        Button b7 = new Button("CANCEL");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        add(b7);

        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
```

```
        new MyFrame2();
    }
}
```

```
import java.awt.*;
class MyFrame3 extends Frame
{
    MyFrame3()
    {
        setLayout(null);

        Button b1 = new Button("OK");
        Button b2 = new Button("CANCEL");
        Button b3 = new Button("FIRSTHELP");
        Button b4 = new Button("CANCEL");
        Button b5 = new Button("FIRSTHELP");
        Button b6 = new Button("CANCEL");
        Button b7 = new Button("CANCEL");
        b1.setBounds(30,50,80,90);
        add(b1);

        b2.setBounds(50,60,120,130);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        add(b7);

        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
}
```

```
public static void main(String[] args)
{
    new MyFrame3();
}
}
```

```
import java.util.*;
class MyStrTokenizer
{
    // "I am working in activenet as a Faculty."
    // "I am residing at SRnagar.";
    public static void main(String[] args) throws Exception
    {
        String str="My name is rami reddy.";
        StringTokenizer st=
            new StringTokenizer(str);
        int i=st.countTokens();
        System.out.println("count="+i);
        while(st.hasMoreTokens())
        {
            try
            {
                String key =st.nextToken();
                System.out.println(key);
            }
            catch(NoSuchElementException e)
            {
            }
        }
    }
}
```

```
import java.awt.*;
import java.awt.event.*;

class MyWindow extends Frame implements WindowListener
{
    MyWindow()
    {
        setSize(300,400);
        show();
        addWindowListener(this);
    }
    public void windowOpened(WindowEvent we)
    { }
    public void windowActivated(WindowEvent we)
    {
        System.out.println("window activated");
    }
    public void windowDeactivated(WindowEvent we)
    { }
    public void windowIconified(WindowEvent we)
    {
        System.out.println("window is iconified");
    }
    public void windowDeiconified(WindowEvent we)
    {
        System.out.println("window deiconified");
    }
    public void windowClosed(WindowEvent we)
    { }
    public void windowClosing(WindowEvent we)
    { }

    public static void main(String[] args)
```

```
{  
    new MyWindow();  
}  
}
```

```
import java.awt.*;  
import java.awt.event.*;  
  
class MyWindow1 extends Frame  
{  
    MyWindow1()  
    {  
        setSize(300,400);  
        show();  
        //MyAdapter ma = new MyAdapter();  
        //addWindowListener(ma);  
        addWindowListener(new MyAdapter());  
  
    }  
    public static void main(String[] args)  
    {  
        new MyWindow1();  
    }  
}  
class MyAdapter extends WindowAdapter  
{  
  
    public void windowActivated(WindowEvent we)  
    {
```

```
        System.out.println("window activated");
    }
}
```

```
class nestedtry
{
    public static void main(String[] args)
    {
        try
        {
            int a=args.length;
            int b=10;
            int c=b/a;
            System.out.println("c value is "+c);
            try
            {
                if(a==1)
                    a=a/(a-a);
                if(a==2)
                {
                    int d[]={10,20};
                    d[5]=50;
                }
            }
            catch(ArithmeticException e)
            {
                System.out.println(e);
            }
            catch(ArrayIndexOutOfBoundsException e)
```

```

        {
            System.out.println(e);
        }
    }
catch(Exception e)
{
    System.out.println("control came to outer catch block");
    System.out.println(e);
}
//main
}//class

```

```

class NewThread implements Runnable
{
    public void run()
    { }

    public static void main(String[] args)
    {
        NewThread nt = new NewThread();
        Thread t = new Thread(nt,"first");
        System.out.println(t.isAlive());
        System.out.println("is the thread is daemon"+t.isDaemon());

        System.out.println("is the thread is daemon"+t.isDaemon());

        t.start();
        t.setDaemon(true);
        System.out.println(t.isAlive());

        System.out.println(t);
        System.out.println(t.getPriority());

        t.setPriority(Thread.MIN_PRIORITY+2);
        System.out.println(t.getPriority());
    }
}

```

```
}
```

```
class outer
{
    int a=10;
    static int b=20;
    class inner
    {
        int c=30;
        //static int d=40;
    }
}
class nonstaticinner
{
    public static void main(String[] args)
    {
        outer ou = new outer();
        System.out.println("a value is "+ou.a);
        System.out.println("b value is :" +outer.b);

        //outer.inner in = new outer().new inner();
        outer.inner in = ou.new inner();
        System.out.println("c value is :" +in.c);
    }
}
```

```
package P1;

class One
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}

import java.io.*;

class onlydir
{
    public static void main(String[] args)
    {
        File f = new File("/work6pm");

        FilenameFilter x = new onlyext(args[0]);
        String s[] = f.list(x);
        for(int i=0;i<s.length;i++)
        {
            System.out.println(s[i]);
        }
    }
}
```

```
        }  
    }
```

```
class onlyext implements FilenameFilter  
{  
    String ext;  
    onlyext(String s)  
    {  
        ext=". "+s;  
    }  
    public boolean accept(File d, String name)  
    {  
        return name.endsWith(ext);  
    }  
}
```

```
class A  
{
```

```
void show()
{
    System.out.println("A class method");
}
void display()
{
    System.out.println("class A display method");
}
}
class B extends A
{
    void show()
    {
        System.out.println("B class method");
    }
}

class override
{
    public static void main(String[] args)
    {
        A a1 = new A();
        B b1 = new B();

        System.out.println(a1);
        System.out.println(b1);

        A x;
        x=a1;
        System.out.println(x);
        System.out.println(a1);
        x.show();

        x=b1;
        System.out.println(x);
        x.show();
    }
}
```

```
class A
{
    int a,b;
    void sum()
    {
        a=10;
        b=20;
        int c=a+b;
        System.out.println("the sum is :" +c);
    }
}
class B extends A
{
    int x,y;
    void sum(int x,int y)
    {
        this.x=x;
        this.y=y;
        int z=x+y;
        System.out.println("the sum is :" +z);
    }
}
class overridedemo
{
    public static void main(String args[])
    {
        B b1 = new B();
        b1.sum();
        b1.sum(50,60);
    }
}
```

```
import java.io.*;  
  
class pipedemo extends Thread  
{  
    static PipedReader pr;  
    static PipedWriter pw;  
  
    pipedemo(String name)  
    {  
        super(name);  
    }  
    public static void main(String[] args) throws IOException  
    {  
        pipedemo pd1 = new pipedemo("src");  
        pipedemo pd2 = new pipedemo("dst");  
        pw = new PipedWriter();  
        pr = new PipedReader(pw);  
  
        pd1.start();  
        pd2.start();  
  
        try  
        {  
            Thread.sleep(2000);  
        }  
        catch(Exception e)  
        { }  
    }  
    public void run()  
    {  
        int j;  
        try  
        {  
            if(getName().equals("src"))
```

```

        {
            for(int i=0;i<10;i++)
                pw.write(getName()+i);
        }
        else
            while((j=pr.read())!=-1)
                System.out.print((char)j);
    }
    catch(Exception e)
    { }
}
}

```

```

import java.awt.*;
import java.awt.event.*;

class popupexample extends Frame implements MouseListener
{
    PopupMenu pm;
    MenuItem mi1,mi2,mi3,mi4;
    popupexample()
    {
        pm = new PopupMenu("a sample popup.....");
        mi1= new MenuItem("one");
        mi2= new MenuItem("two");
        mi3= new MenuItem("three");
        mi4= new MenuItem("four");

        pm.add(mi1);
        pm.add(mi2);
        pm.add(mi3);
        pm.add(mi4);
        add(pm);
        addMouseListener(this);
        setVisible(true);
        setSize(200,200);
        addWindowListener(new WindowAdapter()

        {

    public void windowClosing(WindowEvent we)

```

```
{  
    System.exit(0);  
}  
});  
}  
public void mouseEntered(MouseEvent me)  
{ }  
public void mouseExited(MouseEvent me)  
{ }  
public void mouseClicked(MouseEvent me)  
{ }  
public void mouseReleased(MouseEvent me)  
{ }  
public void mousePressed(MouseEvent me)  
{  
    pm.show(me.getComponent(),me.getX(),me.getY());  
}  
public static void main(String args[])  
{  
    popupexample pp = new popupexample();  
}
```

```
class Q
{
    int n;
    synchronized int get()
    {
        System.out.println("got"+n);
        return n;
    }
    synchronized void put(int n)
    {
        this.n=n;
        System.out.println("put"+n);
    }
}
class producer implements Runnable
{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class consumer implements Runnable
```

```
{  
    Q q;  
    consumer(Q q)  
    {  
        this.q=q;  
        new Thread(this,"consumer").start();  
    }  
    public void run()  
    {  
        int i=0;  
        while(true)  
        {  
            q.get();  
        }  
    }  
}  
class procon  
{  
    public static void main(String[] args)  
    {  
        Q q = new Q();  
        new producer(q);  
        new consumer(q);  
        System.out.println("press ctrl -c to stop");  
    }  
}
```

The producer and consumer example with modified version

---

```
class Q
{
    int n;
    boolean valueset=false;
    synchronized int get()
    {
        if(!valueset)
            try
            {
                wait();
            }
            catch(Exception e)
            {
                System.out.println("InterruptedException");
            }
        System.out.println("got"+n);
        valueset=false;
        notify();
        return n;
    }
    synchronized void put(int n)
    {
        if(valueset)
            try
            {
                wait();
            }
            catch(Exception e)
            {
                System.out.println("InterruptedException");
            }
        this.n=n;
        valueset=true;
        System.out.println("put"+n);
        notify();
    }
}
```

class producer implements Runnable

```

{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class consumer implements Runnable
{
    Q q;
    consumer(Q q)
    {
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.get();
        }
    }
}
class proconfixed
{
    public static void main(String[] args)
    {
        Q q = new Q();
        new producer(q);
        new consumer(q);
        System.out.println("press ctrl -c to stop");
    }
}

```

```
import java.util.*;  
  
class PropertiesDemo  
{  
    public static void main(String[] args)  
    {  
        Properties states = new Properties();  
        String str;  
        states.put("Andhra Pradesh", "Hyderabad");  
        states.put("Tamil Nadu", "Chennai");  
        states.put("Karnataka", "Bangalore");  
        states.put("Kerala", "Trivendram");  
        Set capitals=states.keySet();  
        Iterator i=capitals.iterator();  
        while(i.hasNext())  
        {  
            str=(String)i.next();  
            System.out.println("the capital of " +str+" is  
"+states.getProperty(str));  
        }  
    }  
}
```

```
import java.util.*;
import java.io.*;
class PropertiesDemo1
{
    public static void main(String[] args) throws Exception
    {
        FileOutputStream fos = new FileOutputStream("abc.pro");

        Properties states = new Properties();
        String str;
        states.put("AndhraPradesh","Hyderabad");
        states.put("Tamilnadu","Chennai");
        states.put("Karnataka","Bangalore");
        states.put("Kerala","Trivendram");
        states.store(fos,"c");

        Set capitals=states.keySet();
        Iterator i=capitals.iterator();
        while(i.hasNext())
        {
            str=(String)i.next();
            System.out.println("the capital of " +str+" is
"+states.getProperty(str));
        }
        FileInputStream fis = new FileInputStream("abc.pro");
        states.load(fis);
        states.list(System.out);
    }
}
```

```
class A
{
    public int x;
    protected int y;
    A()
    { }
}
class B extends A
{
    void setdata()
    {
        x=10;
        y=20;
    }
    void display()
    {
        System.out.println("x value is:"+x);
        System.out.println("y value is:"+y);
    }
}
class C extends B
{
    static C c2 = new C();

    void show()
    {
        System.out.println("x value is:"+x);
        System.out.println("y value is:"+y);
    }
}
class protecteddemo
{
    public static void main(String[] args)
    {
        final C c1 = new C();
        c1.setdata();
        c1.show();
        c1.display();
    }
}
```

```
}
```

```
import java.io.*;  
  
class rafdemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        RandomAccessFile raf = new RandomAccessFile("abc.txt","rw");  
        int k;  
        //while((k=raf.read())!=-1)  
        //{
/>            //k=raf.read();  
            //System.out.print((char)k);  
        //}  
  
        raf.seek(15);  
        System.out.println();  
  
        while((k=raf.read())!=-1)  
        {  
            System.out.print((char)k);  
        }  
        raf.writeBytes("corejava class");  
    }  
}
```

```
// System.in -----> standard input stream which is associated with keyboard  
// system.out -----> standard output stream which is associated with monitor or console  
// system.err -----> standard error stream which is associated with console
```

```
//program to read a character from keyboard  
class read  
{  
    public static void main(String[] args) throws Exception  
    {  
        System.out.println("enter a value");  
        int ch=System.in.read();  
        System.out.println("the character is :"+(char)ch);  
    }  
}
```

```
// reading a line from keyboard  
class read2  
{  
    public static void main(String[] args) throws Exception  
    {  
        System.out.println("enter a line");  
        int ch;  
        do  
        {  
            ch=System.in.read();  
            System.out.print((char)ch);  
        }  
        while(ch!='q');  
    }  
}
```

```
// reading string from the keyboard
import java.io.*;

class read3
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter a string");
        DataInputStream dis = new DataInputStream(System.in);
        //String s = dis.readLine();
        //System.out.println("the string is"+s);
        int k;
        do
        {
            k = dis.read();
            System.out.print((char)k);
        }
        while(k!='q');

    }
}
```

```
//reading a line from keyboard using character streams
import java.io.*;

class read4
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter a line");
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String s = br.readLine();
        System.out.println("The string is:"+s);
    }
}
```

```
// Reading a double or int from keyboard
import java.io.*;

class read5
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter an integer!");
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String s=br.readLine();
        int x = Integer.parseInt(s);
```

```
        System.out.println(x+50);
    }
}

import java.io.*;
class readerstream
{
    public static void main(String[] args)
    {
        FileWriter fw=null;
        FileReader fr = null;
        try
        {
            if(args.length!=2)
            {
                System.out.println("invalid arguments");
                return ;
            }
            fw = new FileWriter(args[0].trim());
            fr= new FileReader(args[1].trim());
            int k=0;
            while((k=fr.read())!=-1)
            {
                fw.write(k);
                System.out.print((char)k);
            }
        } catch(Exception e)
        {
            e.printStackTrace();
        } finally
        {
            try
            {
                fw.close();
                fr.close();
            } catch(Exception e)
            {
                System.out.println(e);
            }
        }
    }
}
```

```
        }
    }
}

import java.io.*;

class ReadExternal
{
    public static void main(String[] args)
    {
        FileInputStream fis = null;
        ObjectInputStream ois = null;
        try
        {
            fis=new FileInputStream("external.dat");
            ois=new ObjectInputStream(fis);
            while(true)
            {
                try
                {
                    StudentExternal se = (StudentExternal)ois.readObject();
                    se.print();
                }
                catch(Exception e)
                {
                    break;
                }
            }
            ois.close();
            fis.close();
        }
        catch(Exception ee)
        {
            System.out.println(ee.toString());
        }
    }
}
```

```
class RelatedThread extends Thread
{
    public void run()
    {
        for(int i=0;i<50;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        RelatedThread r1 = new RelatedThread();
        RelatedThread r2 = new RelatedThread();

        r1.start();
        r2.start();
    }
}
```

```
class RelatedThread2 implements Runnable
{
    public synchronized void run()
    {
        System.out.println("thread invoked");
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        RelatedThread2 r1 = new RelatedThread2();
        RelatedThread2 r2 = new RelatedThread2();
```

```
    Thread t1 = new Thread(r1,"first");
    Thread t2 = new Thread(r1,"second");

    Thread t3 = new Thread(r2,"third");
    Thread t4 = new Thread(r2,"forth");

    t1.start();
    t2.start();
    t3.start();
    t4.start();

}
```

```
//:Rethrowing.java
// Demonstrating fillInStackTrace()

class Rethrowing
{

    public static void f() throws Exception
    {
        System.out.println("originating the exception in f()");
        throw new Exception("thrown from f()");
    }

    public static void g() throws Throwable
    {
        try
        {
            f();
        }
        catch(Exception e)
        {
            System.err.println("Inside g(), e.printStackTrace()");
            e.printStackTrace();
            throw e;
        }
    }
}
```

```
public static void main(String[] args) throws Throwable
{
    try
    {
        g();
    }
    catch(Exception e)
    {
        System.err.println("Caught in main, e.printStackTrace()");
        e.printStackTrace();
    }
}
```

```
import java.awt.*;
import java.awt.event.*;

class RubberLine extends Frame
{
    Point start = new Point();
    Point end = new Point();
    RubberLine()
    {
        setSize(300,300);
        addMouseListener(new MouseAdapter()
        {
            public void mousePressed(MouseEvent e)
            {
                start.x=e.getX();
                start.y = e.getY();
            }

            public void mouseReleased(MouseEvent e)
            {

```

```
end.x = e.getX();

end.y = e.getY();

repaint();

}

addMouseMotionListener(new MouseMotionAdapter()

{

    public void mouseDragged(MouseEvent e)

    {

        end.x = e.getX();

        end.y = e.getY();

        repaint();

    }

);

setVisible(true);

}

public void paint(Graphics g)

{

    g.drawLine(start.x,start.y,end.x,end.y);

}

public static void main(String args[])

{

    new RubberLine();

}

}
```

```
class SecondThread implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        SecondThread st = new SecondThread();
        Thread t = new Thread(st);
        t.start();
    }
}
```

```
import java.io.*;
class sequencedemo
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis1 = new FileInputStream("abc.txt");
        FileInputStream fis2 = new FileInputStream("xyz.txt");

        SequenceInputStream sis = new SequenceInputStream(fis1,fis2);

        int k;
        while((k=sis.read())!=-1)
        {
            System.out.print((char)k);
        }
    }
}
```

```

}

class student
{
    int sno;
    int m1,m2,m3;
    public student(int sno,int m1,int m2,int m3){
        this.sno=sno;
        this.m1=m1;
        this.m2=m2;
        this.m3=m3;
    }
    void modify(){
        m1++;m2++;m3++;
    }
    void display(){
        System.out.println("sno= "+sno+" m1="+m1+" m2="+m2+" m3="+m3);
    }
}

class result
{
    int sno;
    String result;
    public result(int sno,String result){
        this.sno=sno;
        this.result=result;
    }
    void modify(String ne){
        result=ne;
    }
    void display(){
        System.out.println("sno="+sno+" result="+result);
    }
}

```

```

class mytest implements Cloneable {

    student stud;
    result res;
    public mytest(int sno,int m1,int m2,int m3,String r){
        stud=new student(sno,m1,m2,m3);

```

```
        res=new result(sno,r);
    }
    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}

class test{
    public static void main(String args[]) throws Exception {
        mytest t=new mytest(101,67,78,89,"first");
        mytest t1=(mytest)t.clone();
        t.stud.display();
        t.res.display();

        t1.stud.display();
        t1.res.display();

        t.stud.modify();
        t.res.modify("destention");
        System.out.println("after modifictation ");
        t.stud.display();
        t.res.display();
        t1.stud.display();
        t.res.display();
    }
}
```

```

abstract class shape
{
    abstract void area();
    void behavior()
    {
        System.out.println("this example show hierarchical inheritance");
    }
}

class Rect extends shape
{
    int length,breadth;
    void getRect()
    {
        length=20;
        breadth=30;
    }
    void area()
    {
        int z=length * breadth;
        System.out.println("the area of reactangle is :" +z);
    }
}

class square extends shape
{
    int a;
    void getsquare()
    {
        a=40;
    }
    void area()
    {
        int x=a * a;
        System.out.println("the area of square is :" +x);
    }
}

class shapedemo
{
    public static void main(String[] args)

```

```

{
    //shape s = new shape();
    //s.area();

    Rect r1 = new Rect();
    r1.getRect();
    r1.area();
    r1.behavior();

    square sq=new square();
    sq.getsquare();
    sq.area();
    sq.behavior();

}

```

We can pass parameter to the main method

---

```

class Simple
{
    public static void main(String args[])
    {
        System.out.println("original main method");
        int x=10;
        main(x);
    }
    public static void main(int x)
    {
        System.out.println("x value is :" +x);
    }
}

```

```
class Simple
{
    void display()
    {
        System.out.println("hello");
    }
}
class simpledemo
{
    public static void main(String[] args)
    {
        Simple s[]=new Simple[5];
        s[0]=new Simple();
        s[0].display();
    }
}
```

```

class sleeptest extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<10;i++)
            {
                Thread.sleep(1000);
                System.out.println("i value is :"+i);
            }
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void main(String[] args)
    {
        sleeptest s1 = new sleeptest();
        sleeptest s2 = new sleeptest();

        s1.start();
        s2.start();
    }
}

class sortarray
{
    public static void main(String[] args)
    {
        int a[]={50,30,20,60,45};

        int n=a.length;

        for(int i=0;i<n;i++)
        {

```

```
        for(int j=i+1;j<n;j++)
        {
            int temp;
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    for(int i=0;i<n;i++)
    {
        System.out.println(a[i]);
    }
}
```

```
import java.util.*;
class sorting
{
    public static void main(String args[])
    {
        Arrays.sort(args);
        for(int i=0;i<args.length;i++)
        {
            System.out.println(args[i]);
        }
    }
}
```

```
import java.util.*;
class stackdemo
{
    public static void main(String []a)
    {
        Stack s=new Stack();
        s.push(new Integer(33));
        s.push(new String("java"));
        s.push(new Float(45.67));
        s.push(new Integer(67));
        s.push(null);
        s.push(null);

        System.out.println(s);
        Object o=s.peek();

        System.out.println("PEEK "+o.toString());
        System.out.println(s);
        Object o1=s.pop();
        System.out.println("after deletion"+o1.toString());
        System.out.println(s);
    }
}

class StackTrace
{
    IllegalArgumentException ex;
    public static void main(String[] argv)
    {
        StackTrace st = new StackTrace();
        st.makeit();
        System.out.println("CONSTRUCTED BUT NOT THROWN");
        st.ex.printStackTrace();
        st.throwit();
    }
}
```

```

// MAY BE NOTREACHED - THINK ABOUT IT!
System.out.println("CONSTRUCTED BUT NOT THROWN");
st.ex.printStackTrace();
}

public void makeit() {
    ex = new IllegalArgumentException("Don't like the weather today");
}
public void throwit() throws IllegalArgumentException {
    throw ex;
}

```

```

class AA
{
    static void classmethod()
    {
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}
class staticoverride
{

```

```
public static void main(String args[])
{
    AA a1 = new AA();
    BB b1 = new BB();

    AA x;
    x=a1;
    x.classmethod();
    x.instancemethod();

    x=b1;
    x.classmethod();
    x.instancemethod();
}
```

```
class ABCD
{
    static void one()
    {
        System.out.println("class ABCD one method");
    }
    static void one(int x)
    {
        System.out.println("x vlaue is :" +x);
    }
}
class XYZ extends ABCD
{ }
class staticinheritdemo
{
    public static void main(String[] args)
    {
        XYZ x1 = new XYZ();
        x1.one();
        x1.one(10);
```

```
    }  
}
```

```
class outer  
{  
    int a=10;  
    static int b=20;  
  
    class inner  
    {  
        int c=30;  
        static int d=40;  
  
        void display()  
        {  
            System.out.println("this is the method in inner class");  
            System.out.println("the outer a is"+b);  
        }  
    }  
}  
  
class StaticInner  
{  
    public static void main(String args[])  
    {  
        outer ou = new outer();  
        System.out.println("a value is :" +ou.a);  
        System.out.println("b value is :" +outer.b);  
  
        outer.inner in = new outer.inner();  
        in.display();  
  
        System.out.println("c value is :" +in.c);  
        System.out.println("d value is :" +outer.inner.d);  
    }  
}
```

```
class outer
{
    int a=10;
    static int b=20;

    class inner
    {
        int c=30;
        //static int d=40;
    }
}

class StaticInner
{
    public static void main(String args[])
    {
        outer ou = new outer();
        System.out.println("a value is :" +ou.a);
        System.out.println("b value is :" +outer.b);

        outer.inner in = new outer.inner();
        in.display();

        System.out.println("c value is :" +in.c);
        System.out.println("d value is :" +outer.inner.d);
    }
}
```

```
class A
{
}
class demo
{
    static A a1 = new A();
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

```
class AA
{
    static void classmethod()
    {
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
```

```
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}

class staticoverride
{
    public static void main(String args[])
    {
        AA a1 = new AA();
        BB b1 = new BB();

        AA x;
        x=a1;
        x.classmethod();
        x.instancemethod();

        x=b1;
        x.classmethod();
        x.instancemethod();
    }
}
```

```
class stopdemo implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is "+i);
            destroy();
        }
    }
    public static void main(String[] args)
    {
        stopdemo st = new stopdemo();
        Thread t = new Thread(st);
        t.start();
    }
}
```

```
class stringdemo
{
    public static void main(String[] args)
    {
        String str1 = new String();
        System.out.println(str1);

        stringdemo sd = new stringdemo();
        System.out.println(sd);
```

```
        }
    }
class stringdemo1
{
    public static void main(String[] args)
    {
        //String str = new
StringBuffer().append("hello").append("World").toString();
        String str="hello"+ "world";
        System.out.println(str);
    }
}
```

```
class StringDemo2
{
    public static void main(String[] args)
    {
        String s1 = new String("hello");
        String s2 = new String("hello");

        String s3 ="java world";
        String s4 =s3;

        String s5 = new String("happy birthday");
        String s6 = new String("Happy Birthday");

        String s9="hai";
        String s10="hai";

        String s11="hello";

// =====
        if(s1.equals("hello"))
            System.out.println("s1 equals hello");
        else
            System.out.println("s1 not equals hello");
//=====
```

```

        if(s1.equals(s2))
            System.out.println("s1 and s2 both are equal");
        else
            System.out.println("both are not equal");
//=====
        if(s1=="hello")
            System.out.println("s1 and hello both references same");
        else
            System.out.println("not same reference");
//=====
        if(s1==s2)
            System.out.println("s1 and s2 references are same");
        else
            System.out.println("s1 and s2 both references are not same");
//=====
        if(s3==s4)
            System.out.println("s3 and s4 references are same");
        else
            System.out.println("s3 and s4 references are not same");
//=====
        if(s9==s10)
            System.out.println("s9 and s10 references are same");
        else
            System.out.println("s9 and s10 references are not same");
//=====
        if(s1==s3)
            System.out.println("s1 and s3 references are same");
        else
            System.out.println("s1 and s3 references are not same");
//=====
        if(s1==s11)
            System.out.println("s1 and s11 references are same");
        else
            System.out.println("s1 and s11 references are not same");
//=====
        if(s5.equalsIgnoreCase(s6))
            System.out.println("s5 and s6 both contents are same");
        else
            System.out.println("s5 and s6 contents are different");
//=====
        String s7="god";
        String s8="good";
        System.out.println("s1 compare to s2 is :" + s1.compareTo(s2));
        System.out.println("s7 compare to s8 is :" + s7.compareTo(s8));
        System.out.println("s8 compare to s7 is :" + s8.compareTo(s7));
//=====

```

```

System.out.println("s1 starts with :" + s1.startsWith("he"));
System.out.println("s1 starts with :" + s1.startsWith("e", 1));
System.out.println("s5 ends with :" + s5.endsWith("day"));
//=====
String str = "satya technologies private limited";
System.out.println("t located at :" + str.indexOf('t'));
System.out.println("t located at :" + str.indexOf('t', 3));
System.out.println("net located at :" + str.indexOf("net"));
System.out.println("net located at :" + str.indexOf("net", 9));
System.out.println("last i located at :" + str.lastIndexOf('i'));
System.out.println("last i located at :" + str.lastIndexOf('i', 10));
System.out.println("last net located at :" + str.lastIndexOf("net"));
//=====
System.out.println("sub string from index 10 to end is :" + str.substring(10));
System.out.println("sub string from index 0 to 9 is :" + str.substring(0, 9));
System.out.println("result of s1.concat(s3) is " + s1.concat(s3));
System.out.println("after concatenation:" + s1);
System.out.println("replace 'T' with 'L' in s1 is :" + s1.replace('T', 'L'));
System.out.println("s1 to Uppercase is :" + s1.toUpperCase());
System.out.println("s6 to lowercase is :" + s1.toLowerCase());
String str1 = " activenet ";
System.out.println("str1 after trim :" + str1.trim());

boolean b = true;
char c = 's';
int i = 9;
double d = 99.99;

System.out.println("boolean value is :" + String.valueOf(b));
System.out.println("char value is :" + String.valueOf(c));
System.out.println("int value is :" + String.valueOf(i));
System.out.println("double value is :" + String.valueOf(d));

System.out.println("the length of s1 is :" + s1.length());
System.out.println("4th char at :" + s1.charAt(3));

System.out.println(s1.hashCode());
System.out.println(s2.hashCode());
}

}

```

```
import java.io.*;

class Student implements Serializable
{
    int sno;
    String sname;
    transient double avg;
    public void getdata(int a,String b,double c)
    {
        sno=a;
        sname=b;
        avg=c;
    }
    public void putdata()
    {
        System.out.println(sno+ " "+sname+ " "+avg);
    }
    public static void main(String[] args) throws Exception
    {

        Student st = new Student();
        st.getdata(101,"ram",55.75);
        FileOutputStream fos = new FileOutputStream("abc.cob");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(st);
        st.putdata();
    }
}
```

```
import java.io.*;

class Student1
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis = new FileInputStream("abc.cob");
        ObjectInputStream ois = new ObjectInputStream(fis);
        Object o= ois.readObject();
        Student st =(Student)o;
        st.putdata();
    }
}
class Student implements Cloneable
{
    int sno;
    String sname;
    public Student(int x,String y)
    {
        sno=x;
        sname=y;
    }
    void display()
    {
        System.out.println("the sno is :" +sno);
        System.out.println("the sname is :" +sname);
    }
    void setname(String name)
    {
        sname=name;
    }
    public static void main(String[] args) throws Exception
    {
        Student s1 = new Student(101,"ram");
        s1.display();
        Student s2=(Student)s1.clone();
        s2.display();
        System.out.println();

        s1.setname("hari");
        s1.display();
    }
}
```

```

        s2.display();

        System.out.println(s1);
        System.out.println(s2);

    }
}

class Studentdemo
{
    public static void main(String[] args) throws MarksOutOfBoundsException
    {
        Student s1 = new Student(101,"ram",99);
        Student s2 = new Student(102,"nikil",120);
        //
        try
        //{
            s1.findResult();
            s2.findResult();
        //}
        //catch(Exception e)
        //{
            //System.out.println(e);
        //
        }

        System.out.println("out of findresult");

    }
}

import java.io.*;

public class StudentExternal implements Externalizable
{
    int sno;
    String sname;
    int m1,m2,m3;
    public StudentExternal()
    { }
}

```

```

public StudentExternal(int sno,String sname,int m1,int m2,int m3)
{
    this.sno=sno;
    this.sname=sname;
    this.m1=m1;
    this.m2=m2;
    this.m3=m3;
}
public void writeExternal(ObjectOutput out)throws IOException
{
    Integer isno= new Integer(sno);
    Integer im1 = new Integer(m1);
    Integer im2 = new Integer(m2);
    Integer im3 = new Integer(m3);
    //writing student no
    out.write(isno.toString().getBytes());
    out.write("\r\n".getBytes());
    //writing student name
    out.write(sname.getBytes());
    out.write("\r\n".getBytes());
    out.write(im1.toString().getBytes());
    out.write("\r\n".getBytes());
    out.write(im2.toString().getBytes());
    out.write("\r\n".getBytes());
    out.write(im3.toString().getBytes());
    out.write("\r\n".getBytes());
}
public void readExternal(ObjectInput in)throws IOException
{
    sno = Integer.parseInt(in.readLine());
    sname = in.readLine();
    m1 = Integer.parseInt(in.readLine());
    m2 = Integer.parseInt(in.readLine());
    m3 = Integer.parseInt(in.readLine());
}
public void print()
{
    System.out.println("sno :" +sno +"\n name
:"+sname+"\nm1:" +m1 +"\nm2" +m2 +"\nm3" +m3);
}
public class subject
{
    String code;
    String title;
}

```

```

int passingmin;
int max;
subject(String cd,String tit,int min,int total)
{
    code =cd;
    title = tit;
    passingmin = min;
    max=total;
}
void displaysubject()
{
    System.out.println("subject code is :" +code);
    System.out.println("name of the subject" +title);
    System.out.println("minimum marks for passing" +passingmin);
    System.out.println("maximum marks are:" +max);
}
public static void main(String[] args)
{
    subject s = new subject("06ct01","java",35,100);
    s.displaysubject();
}
}

```

```

class syn extends Thread
{
    public void run()
    {
        display();
    }
    synchronized void display()
    {
        for(int i=0;i<20;i++)
        {
            System.out.println("i vlaue is :" +i);
        }
    }
    public static void main(String[] args)
    {
        syn s1 = new syn();
    }
}

```

```
        s1.start();
        syn s2 = new syn();
        s2.start();
    }
}
```

```
class syn1 implements Runnable
{
    public void run()
    {
        display();
    }
    public synchronized void display()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                Thread.sleep(1000);
            }
            catch(Exception e)
            {
                System.out.println(Thread.currentThread().getName());
            }
        }
    }
    public static void main(String[] args)
    {
        syn1 s = new syn1();

        Thread t1 = new Thread(s,"first");
        Thread t2 = new Thread(s,"second");

        t1.start();
        t2.start();
    }
}
```

```

class syn2 implements Runnable
{
    int x;
    public static void main(String[] args)
    {
        syn2 s = new syn2();

        Thread t1 = new Thread(s,"first");
        Thread t2 = new Thread(s,"second");

        t1.start();
        t2.start();
    }
    public void run()
    {
        int hold;
        System.out.println("entered into run method");
        System.out.println(Thread.currentThread().getName());
        for(int i=0;i<10;i++)
        {
            synchronized(this)
            {
                System.out.println("entered into syn block");
                System.out.println(Thread.currentThread().getName());
                hold=x+1;
                try
                {
                    Thread.sleep(1000);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
                x=hold;
                System.out.println(Thread.currentThread().getName());
                System.out.println("syn completed");
            }
        }
    }
}

```

```
class Test
{
    public static void main(String args[])
    {
        int a=10;
        int b=0;
        int c=a/b;
        System.out.println("the c value i s:"+c);
    }
}
```

```
import java.io.IOException;

class testExceptions
{
    void method1() throws Throwable
    {
        throw new Throwable("Throwable Exception in method1");
    }
    void method2() throws Throwable
    {
        throw new IOException("Exception in method2");
    }
    try
    {
        method1();
    }
    catch(Throwable th)
    {
        throw th;
        throw th.fillInStackTrace();
    }
}
public static void main(String args[]) throws Throwable
{
```

```
        new testExceptions().method2();
    }
}

import java.awt.*;

class TextDemo extends Frame
{
    TextDemo()
    {
        setLayout(new FlowLayout());
        TextField tf1 = new TextField(10);
        TextField tf2 = new TextField(30);
        tf2.setEchoChar('*');

        TextArea ta = new TextArea(10,20);

        add(tf1);
        add(tf2);
        add(ta);
        setSize(300,400);
        show();
    }
    public static void main(String[] args)
    {
        new TextDemo();
    }
}
```

```
class Third extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                Thread.sleep(1000);
            }
            catch(Exception e)
            {
            }
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        Third t1 = new Third();
        t1.start();
        Third t2 = new Third();
        t2.start();
        Third t3 = new Third();
        t3.start();
    }
}
```

```
class throwsdemo
{
    public static void main(String[] args)
    {
        try
        {
            one();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    static void one()
    {
        try
        {
            two();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    static void two()
    {
        try
        {
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is :" +c);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
class throwsdemo1
{
    public static void main(String[] args) throws Exception
    {
        one();
    }
    static void one()
    {
        two();
    }
    static void two() throws Exception
    {
        int a=10;
        int b=0;
        int c=a/b;
        System.out.println("c value is :" +c);
    }
}
```

```
import java.util.*;

class Treeset
{
    public static void main(String[] args)
    {
        TreeSet t = new TreeSet();
        t.add("20");
        t.add("10");
        t.add("30");
        t.add(null);

        System.out.println(t);
    }
}
```

```
}
```

```
import java.util.*;  
  
class Emp  
{ }  
  
class Student  
{ }  
  
class Salary  
{ }  
  
class VectorDemo  
{  
    public static void main(String[] args)  
    {  
        Vector v = new Vector();  
        System.out.println("capacity is :" + v.capacity());  
        v.add("10");  
        v.add("30");  
        v.add("20");  
        v.add("40");  
        /*  
         Enumeration e = v.elements();  
         while(e.hasMoreElements())  
         {  
             Object o = e.nextElement();  
             System.out.println("the object is :" + o);  
         }  
         Iterator i = v.iterator();  
         while(i.hasNext())  
         {  
             Object o = i.next();  
             System.out.println("the object is :" + o);  
         }  
         */  
        ListIterator li = v.listIterator();  
        while(li.hasNext())  
        {  
            Object o = li.next();  
        }  
    }  
}
```

```

        System.out.println("the object is :" + o);
    }

    while(li.hasPrevious())
    {
        Object o1 = li.previous();
        System.out.println("the object is :" + o1);
    }
}

```

```

import java.io.*;

public class WriteExternal
{
    public static void main(String[] args)
    {
        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            FileOutputStream fos = new FileOutputStream("external.dat");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            while(true)
            {
                System.out.println("enter the number (0 to stop) " );
                int sno = Integer.parseInt(br.readLine());
                if(sno==0)
                    break;
                System.out.print("enter the name");
                String sname=br.readLine();
                System.out.print("enter the mark1 :");
                int m1 = Integer.parseInt(br.readLine());
                System.out.print("enter the mark2 :");
                int m2 = Integer.parseInt(br.readLine());
                System.out.print("enter the mark3 :");
                int m3 = Integer.parseInt(br.readLine());

                StudentExternal st = new
                StudentExternal(sno,sname,m1,m2,m3);
            }
        }
    }
}

```

```
        oos.writeObject(st);
    }
    oos.flush();
    oos.close();
    fos.close();
}
catch(Exception e)
{
    System.out.println(e);
}
}
```

```
class yieldingthread extends Thread
{
    int countdown=6;
    static int threadcount=0;
    yieldingthread()
    {
        super(""+++threadcount);
        start();
    }
    public String toString()
    {
        return "#" + getName() + ":" + countdown;
    }
    public void run()
    {
        while(true)
        {
            System.out.println(this);
            if(--countdown == 0)
                return;
            yield();
        }
    }
}
```

```
        }
    }
public static void main(String[] args)
{
    for(int i=0;i<5;i++)
    {
        new yieldingthread();
    }
}
```

ramireddy.satya

---

---

The following programs are more example programs

---

---

//Demonstration of Bitwise operators

```
class Bitwise
{
    public static void main (String args[])
    {
        int x = 5, y = 6;
        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("x & y = " + (x & y));
        System.out.println("x | y = " + (x | y));
        System.out.println("x ^ y = " + (x ^ y));
    }
}
```

```
//Program to validate the inputted date using if..else construct

public class DateValidation

{

    public static void main(String args[])

    {

        if ( args.length != 3)

        {

            System.out.println("Usage : Java DateValidation <dd> <mm> <yy>");

            System.exit(0);

        }

        int dd, mm, yy;

        dd = Integer.parseInt(args[0]);

        mm = Integer.parseInt(args[1]);

        yy = Integer.parseInt(args[2]);

        if (yy > 1900)

        {

            if ( mm==1||mm==3||mm==5||mm==7||mm==8||mm==10||mm==12)

            {

                if (dd>=1 && dd <= 31)
```

```
{  
    System.out.println("Date is valid");  
    System.exit(0);  
}  
  
else  
{  
    System.out.println("Day is invalid");  
    System.exit(0);  
}  
  
}  
  
else  
{  
    if (mm==4||mm==6||mm==9||mm==11)  
    {  
        if (dd>=1 && dd<=30)  
        {  
            System.out.println("Date is valid");  
            System.exit(0);  
        }  
        else  
{  
            System.out.println("Day is invalid");  
            System.exit(0);  
        }  
    }  
}
```

```
    }

}

else

{

    if (mm==2)

    {

        if ((yy%400)==0)

        {

            if (dd>=1 && dd<=29)

            {

                System.out.println("Date is valid");

                System.exit(0);

            }

        }

        else

        {

            System.out.println("Date is invalid");

            System.exit(0);

        }

    }

    else

    {

        if (dd>=1 && dd<=28)

        {

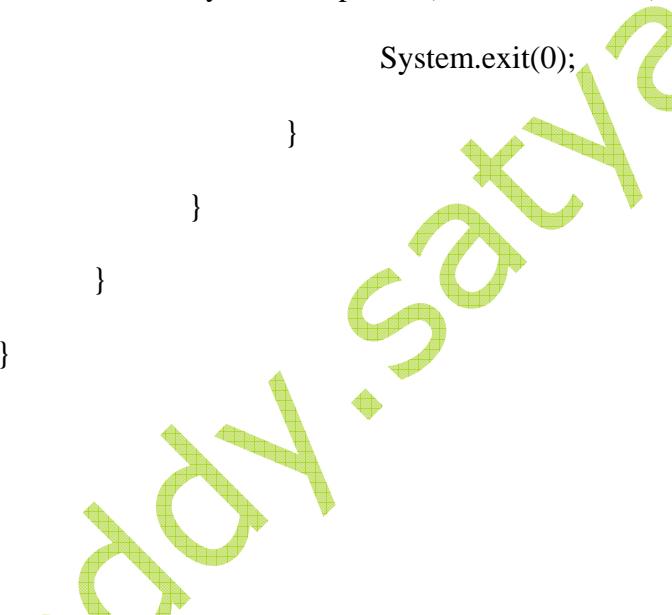
            System.out.println("Date is valid");

        }

    }

}
```

```
        System.exit(0);  
    }  
    else  
    {  
        System.out.println("Date is invalid");  
        System.exit(0);  
    }  
}  
}  
}  
}  
}
```



```
/* calculate volume of box */

class Box2
{
    double width;
    double height;
    double depth;
}

class Second
{
    public static void main(String[] args)
    {
        Box2 mybox=new Box2();
        double vol;
        mybox.width=10;
        mybox.height=20;
        mybox.depth=15;
        vol=mybox.width*mybox.height*mybox.depth;
        System.out.println("volume is "+vol);
    }
}
```

```
class Box4
{
    double width;
    double height;
    double depth;
    void volume()
    {
        double vol=width*height*depth;
        System.out.println("volume is "+vol);
    }
}
class Fourth
{
    public static void main(String[] args)
    {
        Box4 mybox=new Box4();
        Box4 mybox1=new Box4();
        mybox.width=10;
        mybox.height=20;
        mybox.depth=15;
        mybox1.width=5;
        mybox1.height=6;
        mybox1.depth=5;
        mybox.volume();
        mybox1.volume();
    }
}
```

```
class Box5
{
    double width;
    double height;
    double depth;
    double volume()
    {
        return width*height*depth;
    }
}
class Fifth
{
    public static void main(String[] args)
    {
        Box5 mybox=new Box5();
        Box5 mybox1=new Box5();
        double vol;
        mybox.width=10;
        mybox.height=20;
        mybox.depth=15;
        mybox1.width=5;
        mybox1.height=6;
        mybox1.depth=5;
        vol=mybox.volume();
        System.out.println("volume is "+vol);
        vol=mybox1.volume();
        System.out.println("volume is "+vol);
    }
}//Calculate the Volume of Two Boxes
```

```
// HelloWorld Program

class HelloWorld
{
    public static void main(String[] a)
    {
        System.out.println("Hello World!");
    }
}
```

```
// Demonstration of increment (++) and decrement (--) operators  
class IncrementDecrement
```

```
{
```

```
    public static void main (String args[])
```

```
{
```

```
    int x = 10, y = 20;
```

```
    System.out.println("x = " + x);
```

```
    System.out.println("y = " + y);
```

```
    System.out.println("++x = " + ++x);
```

```
    System.out.println("y++ = " + y++);
```

```
    System.out.println("x = " + x);
```

```
    System.out.println("y = " + y);
```

```
}
```

```
}
```

```
//Demonstrating use of static variables
```

```
public class Pilot
```

```
{
```

```
    static int speed;
```

```
    String pName;
```

```
    public static void increaseSpeed()
```

```
{
```

```
        if ( speed > 250 )
```

```
{
```

```
            speed = speed;
```

```
}
```

```
        else
```

```
{
```

```
            speed += 50;
```

```
}
```

```
}
```

```
    public static void initSpeed()
```

```
{  
    speed=60;  
}  
  
public static void decreaseSpeed()  
{  
    if ( speed < 60)  
    {  
        speed=60;  
    }  
    else  
    {  
        speed -= 50;  
    }  
}  
  
public void showSpeed()  
{  
    System.out.println("Speed is : " + speed);  
}  
  
public static void main (String args[])  
{
```

```
int ch=0;

Pilot p1=new Pilot();

p1.pName = "Keyur";

Pilot p2=new Pilot();

p2.pName = "Mayur";

p1.initSpeed();

p1.increaseSpeed();

p1.showSpeed();

p1.increaseSpeed();

p2.showSpeed();

p2.decreaseSpeed();

p1.showSpeed();

}

}
```

```
class Box7
{
    double width;
    double height;
    double depth;
    Box7(double w,double h,double d)
    {
        width=w;
        height=h;
        depth=d;
    }
    double volume()
    {
        return width*height*depth;
    }
}
class Seventh
{
    public static void main(String[] args)
    {
        Box7 mybox=new Box7(5,5,5);
        Box7 mybox1=new Box7(6,6,6);
        double vol;
        vol=mybox.volume();
        System.out.println("volume is "+vol);
        vol=mybox1.volume();
        System.out.println("volume is "+vol);
    }
}
```

```
//Demonstration of Shift operators (<< >> >>>)

class Shift

{

    public static void main (String args[])

    {

        int x = 7;

        System.out.println("x = " + x);

        System.out.println("x >> 2 = " + (x >> 2));

        System.out.println("x << 1 = " + (x << 1));

        System.out.println("x >>> 1 = " + (x >>> 1));

    }

}
```

The output :

```
x = 7
x >> 2 = 1
x << 1 = 14
x >>> 1 = 3
```

```
//finding volume of Boxes using constructors
class Box6
{
    double width;
    double height;
    double depth;
    Box6()
    {
        width=10;
        height=20;
        depth=15;
    }
    double volume()
    {
        return width*height*depth;
    }
}
class Sixth
{
    public static void main(String[] args)
    {
        Box6 mybox=new Box6();
        Box6 mybox1=new Box6();
        double vol;
        vol=mybox.volume();
        System.out.println("volume is "+vol);
        vol=mybox1.volume();
        System.out.println("volume is "+vol);
    }
}
```

/Program to generate 10 random numbers, sort them, and display

```
//SortNumbers.java
import java.util.*;

public class SortNumbers
{
    public static void main(String args[])
    {
        int numbers[] = new int[10];
        int Ascending[] = new int[10];

        Random rnd = new Random();

        for( int i=0; i< numbers.length; i++)
        {
            numbers[i] = rnd.nextInt(100);
        }

        for( int i=0; i< numbers.length; i++)
        {
            Ascending[i] = numbers[i];
        }

        for( int i=0; i < Ascending.length; i++)
        {
            for ( int j=i+1; j< Ascending.length; j++)
            {
                int temp;
                if (Ascending[i] > Ascending[j])
                {
                    temp = Ascending[i];
                    Ascending[i] = Ascending[j];
                    Ascending[j] = temp;
                }
            }
        }

        System.out.println("Original\t\tRearranged");
    }
}
```

```
for( int i=0; i < numbers.length ; i++)
{
    System.out.println(numbers[i] +"\t\t"+Ascending[i]);
}
}
```

ramireddy.satya

```
//      Program to test Command-line Arguments

TestArgs.java
public class TestArgs
{
    public static void main(String args[])
    {
        if(args.length > 0)
        {
            for(int i=0; i < args.length ; i++)
            {
                System.out.println("Argument # " + i + " is :" + args[i]);
            }
        }
        else
        {
            System.out.println("You did not provide any arguments");
        }
    }
}
```

```
//Calculate the volume of 2 BOXs
class Box3
{
    double width;
    double height;
    double depth;
}
class Third
{
    public static void main(String[] args)
    {
        Box3 mybox=new Box3();
        Box3 mybox1=new Box3();
        double vol;
        mybox.width=10;
        mybox.height=20;
        mybox.depth=15;
        mybox1.width=5;
        mybox1.height=6;
        mybox1.depth=5;
        vol=mybox.width*mybox.height*mybox.depth;
        System.out.println("volume is "+vol);
        vol=mybox1.width*mybox1.height*mybox1.depth;
        System.out.println("volume is "+vol);
    }
}
```

```
class bitlogic
{
    public static void main(String[] args)
    {
        int a=3;
        int b=6;
        int c=a|b;
        int d=a&b;
        int e=a^b;
        int f=(~a&b)|(a&~b);
        System.out.println("the value of c="+c);
        System.out.println("the value of d="+d);
        System.out.println("the value of e="+e);
        System.out.println("the value of f="+f);
    }
}
```

```
class leftshift
{
    public static void main(String[] args)
    {
        int i;
        byte b,c,a=49;
        c=(byte)(a<<2);
        i=a<<2;
        System.out.println("the original value of a is "+a);
        System.out.println("the value of i is "+i);
        System.out.println("the value of c is "+c);
    }
}
```

```
class promote
{
    public static void main(String[] args)
    {
        byte b=42;
        char c='a';
        short s=1024;
        int i=5000;
        float f=5.67f;
        double d=.1234;
        double result=(f*b)+(i/c)-(d*s);
        System.out.println("result is "+result);
    }
}
```

```
class rightshift
{
    public static void main(String[] args)
    {
        int i;
        byte b,c,a=64;
        c=(byte)(a>>2);
        i=a>>2;
        System.out.println("the original value of a is "+a);
        System.out.println("the value of i is "+i);
        System.out.println("the value of c is "+c);
    }
}
```

```
class unsignedrightshift
{
    public static void main(String[] args)
    {
        int i;
        byte b,c,a=-1;
        c=(byte)(a>>2);
        i=a>>24;
        System.out.println("the original value of a is "+a);
        System.out.println("the value of i is "+i);
        System.out.println("the value of c is "+c);
    }
}
```

## If-else conditions

```
class if5
{
    public static void main(String[] arg)
    {
        int a,b,c,big;
        a=Integer.parseInt(arg[0]);
        b=Integer.parseInt(arg[1]);
        c=Integer.parseInt(arg[2]);
        if(a>b)
        {
            if(a>c)
            {
                big=a;
            }
            else
            {
                big=c;
            }
        }
        else
        {
            if(b>c)
            {
                big=b;
            }
            else
            {
                big=c;
            }
        }
        System.out.println("the biggest number is"+big);
    }
}
```

```
class if1
{
    public static void main(String[] args)
    {
        int i=100;
        if(i==100)
            System.out.println("you have scored century");
    }
}
```

```
class if4
{
    public static void main(String[] arg)
    {
        float bs,hr,da,gs;
        System.out.println("\n Enter the basic salary");
        bs=Integer.parseInt(arg[0]);
        if (bs>5000)
        {
            da=bs*10/100;
            hr=bs*20/100;
        }
        else
        {
            da=bs*5/100;
            hr=bs*15/100;
        }
        gs=bs+da+hr;
        System.out.println("the gross salary is "+gs);
    }
}
```

```
class if2
{
    public static void main(String[] args)
    {
        int i=100;
        if(i==100)
            System.out.println("you have scored century");
        else
            System.out.println("you have not scored century");
    }
}
```

```
class if6
{
    public static void main(String[] arg)
    {
        int salary,intrest,tax;
        salary=Integer.parseInt(arg[0]);
        if(salary>1000 && salary<2000)
        {
            intrest=5;
            tax=salary*intrest/100;
            System.out.println("the tax is "+tax);
        }
        else if(salary>2000 && salary<3000)
        {
            intrest=10;
            tax=salary*intrest/100;
            System.out.println("the tax is "+tax);
        }
        else if(salary>3000 && salary<4000)
        {
            intrest=15;
            tax=salary*intrest/100;
            System.out.println("the tax is "+tax);
        }
    }
}
```

```
}

class if3
{
    public static void main(String[] args)
    {
        int n=20;
        if(n%2==0)
        {
            System.out.println("the number is even");
        }
        else
        {
            System.out.println("the number is odd");
        }
    }
}
```

## Switch-case-default

```
class switch1
{
    public static void main(String[] arg)
    {
int i;
i=Integer.parseInt(arg[0]);
switch(i)
{
case 1:
    System.out.println("you have entered 1");
break;
case 2:
    System.out.println("you have entered 2");
break;
case 3:
    System.out.println("you have entered 3");
break;
default:
    System.out.println("enter only 1,2or 3");
}
}
}
```

```
class switch2
{
    public static void main(String[] arg)
    {

String s;
s=arg[0];
char ch[]={s.toCharArray()};
switch(ch[0])
{
case 'a':
    System.out.println("entered a");break;
case 'b':
    System.out.println("entered b");break;
case 'c':
    System.out.println("entered c");break;
case 'd':
    System.out.println("entered d");break;
default:
    System.out.println("entered char is not a,b,c,d");
}
}
}
```

```
class switch3
{
    public static void main(String[] arg)
    {

int choice,n1,n2,res=0;
n1=Integer.parseInt(arg[0]);
n2=Integer.parseInt(arg[1]);
choice=Integer.parseInt(arg[2]);
switch(choice)
{
case 1:
    res=n1+n2;
    break;
case 2:
    res=n1-n2;
    break;
case 3:
    res=n1*n2;
    break;
case 4:
    res=n1/n2;
    break;
default:
    System.out.println("wrong choice");
}
System.out.println("result is "+res);
}
}
```

## While loop

```
class while1
{
    public static void main(String[] arg)
    {

int n;
n=Integer.parseInt(arg[0]);
while(n<=10)
{
System.out.println("\nthe number is "+n);
n=n+1;
}
}
}
```

```
class while2
{
    public static void main(String[] arg)
    {

        int n,sum=0,count=1;
        n=Integer.parseInt(arg[0]);
        while(count<=n)
        {
            // System.out.println("\n the value of count is %d",count);
            //System.out.println("\n the value of sum is %d",sum);
            sum=sum+count;
            count=count+1;
        }
        System.out.println("sum is "+sum);
    }
}
```

```
class while3
{
    public static void main(String[] arg)
    {

        int b,result,n,ch=1,div=2,flag=1;
        n=Integer.parseInt(arg[0]);
        if(n==2)
        {
            flag=1;
        }
        else
        {
            b=n/2;
            while(ch<=b)
            {
                result=n%div;
                if(result==0)
                {
                    System.out.println("the entered number is not prime number:");
                    flag=0;
                    break;
                }
                else
                {
                    div++;
                    ch++;
                }
            }
            if(flag==1)
                System.out.println("the number is prime");
        }
    }
}
```

```
class while4
{
    public static void main(String[] arg)
    {

int ch;
while(true)
{
    System.out.println("\n1.C\n2.C++\n3.JAVA");
    System.out.println("enter ur choice 1/2/3/");
    ch=Integer.parseInt(arg[0]);
switch(ch)
{
    case 1:System.out.println("welcome to C");
    case 2:System.out.println("welcome to C++");
    case 3:System.out.println("welcome to JAVA");
}
}
}
}
```

```
class while5
{
    public static void main(String[] args)
    {

        //while(4<1)
        do
        {
            System.out.println("hello");
        }
        while(4<1);
    }
}
```

```
class A
{
    public String toString()
    {
        return "ramireddy";
    }

    public static void main(String[] args)
    {
        A a1 = new A();
        A a2 = new A();
        System.out.println(a1);
        System.out.println(a2);
    }
}
```

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
```

```
/*
<applet code="ApltoApl" width=400 height=300 name=a1>
<param name=p1 value=a2>
</applet>

<applet code="ApltoApl" width=400 height=300 name=a2>
<param name=p1 value=a1>
</applet>

*/

public class ApltoApl extends Applet implements ActionListener
{
    Button b1;
    String str;
    AppletContext ac;
    ApltoApl a;
    public void init()
    {
        b1 = new Button("red");
        b1.addActionListener(this);
        add(b1);
    }
    public void start()
    {
        ac=getAppletContext();
        a=(ApltoApl)ac.getApplet(getParameter("p1"));
    }
    public void actionPerformed(ActionEvent ae)
    {
        String s = ae.getActionCommand();
        if(s.equals("red"))
            a.setBackground(Color.red);
    }
}
```

applets are basically using for developing the dynamic web pages like DHTML  
applets are always access through internet browsers.(java enabled browsers)  
applets are compiled java programs.  
applets are java programs which can be embedded with html.  
once the applet is loaded into client system it has no connections with server

```
class arr_clone
{
    public static void main(String[] args)
    {
        int arr[]={11,22,33};
        int brr[]=(int[])arr.clone();
        arr[1]=100;
        for(int i=0;i<arr.length;i++)
            System.out.println(arr[i]);

        for(int i=0;i<brr.length;i++)
            System.out.println(brr[i]);
    }
}
```

```
class arraydemo
{
    public static void main(String[] args)
    {
        int a[]={10,20,30,40,50};
        for(int i=0;i<5;i++)
        {
            System.out.println(a[i]);
        }
    }
}
```

```
import java.util.*;  
  
class ArrayListDemo  
{  
    public static void main(String[] args)  
    {  
        ArrayList a1 = new ArrayList(20);  
        System.out.println("the size is "+a1.size());  
        a1.add("10");  
        a1.add("30");  
        a1.add("40");  
        a1.add("20");  
        a1.add("20");  
        a1.add(null);  
        a1.add(null);  
        a1.add(3,"50");  
  
        System.out.println(a1);  
    }  
}
```

```
class assertionpropagate
{
    public static void main(String[] args)
    {
        one();
    }
    public static void one()
    {
        two();
    }
    public static void two()
    {
        three();
    }
    public static void three()
    {
        int a=0;
        int b=0;

        assert a>=10;
        assert b>=10;

        int c=a/b;
        System.out.println("c value is :" +c);
    }
}
```

```
class assertiontest
{
    public static void main(String[] args)
    {
        int a=10;
        int b=0;

        assert a>0;
        assert b>0:"b value is less than zero";

        int c=a/b;

        System.out.println("the c value is"+c);
    }
}
```

```
import java.util.Scanner;
import java.io.IOException;

public class AssertionTest1
{
    public static void main(String argv[]) throws IOException
    {
        Scanner reader = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = reader.nextInt();
        //Assertion.NDEBUG=false;
        Assertion.assert(age>=18, "You are too young to vote");
        // use age
        System.out.println("You are eligible to vote");
    }
}
```

```
class A
{
    public static void main(String[] args)
    {
        System.out.println(args[0]);
        System.out.println(args[1]);
    }
}
class B
{
    public static void main(String[] args)
    {
        String arg[]={ "hai","hello"};
        A.main(arg);
    }
}
```

```
class A
{
    public static void main(String[] args)
    {
        int x=Integer.parseInt(args[0]);
        int y=Integer.parseInt(args[1]);

        int z=x+y;
        System.out.println("the sum is :" +z);
    }
}
```

```
class Bank
{
    public static void main(String[] args) throws InsufficientFundsException
    {
        int bal=1000;
        int withdrawn=2000;
        if(bal<withdrawn)
            throw new InsufficientFundsException("bal is less than
withdrawn");
        else
            System.out.println("proceed with transaction");
    }
}
```

```
class AA
{
    static void classmethod()
    {
```

```
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}

class staticoverride
{
    public static void main(String args[])
    {
        AA a1 = new AA();
        BB b1 = new BB();

        AA x;
        x=a1;
        x.classmethod();
        x.instancemethod();

        x=b1;
        x.classmethod();
        x.instancemethod();
    }
}
```

```
import java.io.*;  
  
class bisdemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        String s = "after completion of corejava everybody trying to attend adv  
java";  
        byte b1[]={s.getBytes()};  
        ByteArrayInputStream bis = new ByteArrayInputStream(b1);  
  
        boolean b2 = bis.markSupported();  
        System.out.println("mark supported! "+b2);  
  
        bis.skip(5);  
  
        int k;  
        while((k=bis.read())!=-1)  
        {  
            System.out.print((char)k);  
        }  
    }  
}
```

```
public class BitwiseOperatorsDemo {  
    public static void main(String args[]){  
        int x = 7;  
        int y = 5;  
        int z;  
        System.out.println("x & y : "+(x & y));  
        System.out.println("x | y : "+(x | y));  
        System.out.println("x ^ y : "+(x ^ y));  
        System.out.println("~x : "+(~x));  
        System.out.println("x << y : "+(x << y));  
        System.out.println("x >> y : "+(x >> y));  
    }  
}
```

```
class box  
{  
    double width;  
    double height;  
    double depth;  
    box()
```

```

{
    width=10.5;
    height=20.5;
    depth=30.5;
}
box(double x,double y,double z)
{
    width=x;
    height=y;
    depth=z;
}
box(double x)
{
    width=height=depth=x;
}
void output()
{
    System.out.println("width is :" + width);
    System.out.println("height is :" + height);
    System.out.println("depth is :" + depth);
}
}
class boxweight extends box
{
    double weight;
    boxweight()
    {
        super();
        weight=80.5;
    }
    boxweight(double p,double q,double r,double s)
    {
        super(p,q,r);
        weight=s;
    }
    boxweight(double x)
    {
        //width=height=depth=weight=x;
        super(x);
        weight=x;
    }
    void display()
    {
        System.out.println("width is :" + width);
        System.out.println("height is :" + height);
        System.out.println("depth is :" + depth);
    }
}

```

```
        System.out.println("weight is :" + weight);
    }
}
class boxdemo
{
    public static void main(String[] args)
    {
        boxweight b1 = new boxweight();
        boxweight b2 = new boxweight(55.5,66.6,77.7,88.8);
        boxweight b3 = new boxweight(99.9);

        b1.display();
        b2.display();
        b3.display();
    }
}
```

```
//CalendarTest.java
import java.util.*;
public class calendartest
{
    public static void main(String args[])
    {
        GregorianCalendar cal = new GregorianCalendar();
        System.out.println("Day is : " + cal.get(Calendar.DAY_OF_MONTH));
        System.out.println("Month is : " + cal.get(Calendar.MONTH));
        System.out.println("Year is : " + cal.get(Calendar.YEAR));
        if (cal.get(Calendar.ERA) == GregorianCalendar.AD)
            System.out.println("Era is : AD");
        else
            System.out.println("Era is : BC");
        if (cal.isLeapYear(cal.get(Calendar.YEAR)))
            System.out.println("Year is leap");
        else
            System.out.println("Year is non-leap");
    }
}
```

```
import java.awt.*;  
  
class CheckDemo extends Frame  
{  
    CheckDemo()  
    {  
        setLayout(new FlowLayout());  
  
        CheckboxGroup cbg = new CheckboxGroup();  
  
        Checkbox cb1 = new Checkbox("corejava",cbg,false);  
        Checkbox cb2 = new Checkbox("advjava",cbg,true);  
        Checkbox cb3 = new Checkbox("j2EE",cbg,false);  
        add(cb1);  
        add(cb2);  
        add(cb3);  
        setSize(300,400);  
        show();  
    }  
    public static void main(String[] args)  
    {  
        new CheckDemo();  
    }  
}
```

```
import java.awt.*;  
  
class ChoiceDemo extends Frame  
{  
    ChoiceDemo()  
    {  
        setLayout(new FlowLayout());  
        Choice c = new Choice();  
        c.addItem("corejava");  
        c.addItem("adv.java");  
        c.addItem("j2ee");  
        c.addItem(".net");  
  
        add(c);  
        setSize(300,400);  
        show();  
    }  
    public static void main(String[] args)  
    {  
        new ChoiceDemo();  
    }  
}
```

```
import java.util.ArrayList;
```

```
import java.util.Iterator;

class Int {
    private int i;

    public Int(int ii) {
        i = ii;
    }

    public void increment() {
        i++;
    }

    public String toString() {
        return Integer.toString(i);
    }
}

public class Cloning {

    public static void main(String[] args) {
        ArrayList v = new ArrayList();
        for (int i = 0; i < 10; i++)
            v.add(new Int(i));
        System.out.println("v: " + v);
        ArrayList v2 = (ArrayList) v.clone();
        // Increment all v2's elements:
        for (Iterator e = v2.iterator(); e.hasNext();)
            ((Int) e.next()).increment();
        // See if it changed v's elements:
        System.out.println("v: " + v);
    }
} //:~
```

```
// Making a Collection Read-Only
//Making a collection read-only involves wrapping the collection
// in another object whose mutation methods all throw UnsupportedOperationException.

List stuff = Arrays.asList(new String[]{"a", "b"});

// Make a list read-only
List list = new ArrayList(stuff);
list = Collections.unmodifiableList(list);

try {
    // Try modifying the list
    list.set(0, "new value");
} catch (UnsupportedOperationException e) {
    // Can't modify
}

// Make a set read-only
Set set = new HashSet(stuff);
set = Collections.unmodifiableSet(set);

// Make a map read-only
Map map = new HashMap();
// Add key/value pairs ...
map = Collections.unmodifiableMap(map);
```

```
import java.util.*;

class collecreadonly
{
    public static void main(String args[])
    {
```

```
import java.util.*;  
  
ArrayList a1 = new ArrayList();  
a1.add("a");  
a1.add("b");  
a1.add("x");  
a1.add("y");  
  
List li = new ArrayList(a1);  
li=Collections.unmodifiableList(li);  
try  
{  
    li.set(0,"hello");  
}  
catch(UnsupportedOperationException e)  
{  
    System.out.println(e);  
}  
// -----  
  
Vector v1 = new Vector();  
v1.add("a");  
v1.add("b");  
v1.add("x");  
v1.add("y");  
  
List l1 = new Vector(v1);  
l1=Collections.unmodifiableList(l1);  
try  
{  
    l1.set(0,"hello");  
}  
catch(UnsupportedOperationException e)  
{  
    System.out.println(e);  
}  
// List stuff=Arrays.asList(new String[]{"a","b"});  
}  
}
```

```
class Mycomp implements Comparator
{
    public int compare(Object x, Object y)
    {
        String a,b;
        a=(String)x;
        b=(String)y;
        int i,j,k;
        i=a.lastIndexOf(' ');
        j=b.lastIndexOf(' ');
        k=a.substring(i).compareTo(b.substring(j));

        if(k==0)
            return a.compareTo(b);
        else
            return k;
    }
}
class CompDemo1
{
    public static void main(String[] args)
    {
        TreeSet t = new TreeSet(new Mycomp());
        t.add("rami reddy");
        t.add("mohan kishore");
        t.add("hari gupta");
        t.add("nikhil sastry");

        System.out.println(t);
    }
}
```

```
public class CompoundOperatorsDemo {  
    public static void main(String args[]){  
        int x = 0, y = 5;  
        x += 3;  
        System.out.println("x : "+x);  
        y *= x;  
        System.out.println("y : "+y);  
  
        /*Similarly other operators can be applied as shortcuts. Other  
         compound assignment operators include boolean logical  
         , bitwiseand shift operators*/  
    }  
}
```

```
class A
{
    int a,b;
    A(int x,int y)
    {
        a=x;
        b=y;
    }
}
class Con1
{
    public static void main(String[] args)
    {
        A a1=new A();
        A a2=new A(55,66);
    }
}
```

```
class A
{
    int a,b;
```

```
A()
{
    a=10;
    b=20;
}

A(int x,int y)
{
    a=x;
    b=y;
}
A(A x)
{
    a=x.a;
    b=x.b;
}
void output()
{
    System.out.println("a value is "+a);
    System.out.println("b value is "+b);
}
}
class ConstDemo
{
    public static void main(String[] args)
    {
        A a1 = new A();
        A a2 = new A(77,88);
        A a3 =new A(a2);

        a1.output();
        a2.output();
        a3.output();
    }
}
```

```
class Test
{
    int a,b,c;
    Test(int x,int y)
    {
        a=x;
        b=y;
    }
    Test(int x,int y,int z)
    {
        this(x,y);
        c=z;
    }
    void display()
    {
        System.out.println("a value is :" +a+ " " +"b value is :" +b+ " "+"c value is
        :" +c);
    }
}
class constdemo1
{
    public static void main(String[] args)
    {
        Test t1 = new Test(22,33,44);
        t1.display();
    }
}
```

```
class copyarray
{
    public static void main(String[] args)
    {
        int a[]={10,20,30,40};
        int b[] = new int[4];
        b=a;
        for(int i=0;i<b.length;i++)
        {
            System.out.println(b[i]);
        }
    }
}
```

class A

```

{
    synchronized void show(B b)
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+"\tEntered A.show()");
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            System.out.println("An Interruption has Raised");
        }
        System.out.println("Trying to call B.last()");
        System.out.println(Thread.currentThread().getName());
        b.last();
    }
    synchronized void last()
    {
        System.out.println("Inside A.Last");
    }
}
class B
{
    synchronized void display(A a)
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+"\tEntered B.display()");
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            System.out.println("An Exception has Raised");
        }
        System.out.println("Trying to call A.last()");
        System.out.println(Thread.currentThread().getName());
        a.last();
    }
    synchronized void last()
    {
        System.out.println("Inside B.Last");
    }
}
class Deadlock implements Runnable

```

```
{  
    A a = new A();  
    B b = new B();  
    Deadlock()  
    {  
        Thread.currentThread().setName("Main Thread");  
        Thread t=new Thread(this,"child Thread");  
        //new Thread(this,"racing thread").start();  
  
        t.start();  
        a.show(b);  
        System.out.println("Back in Main Thread");  
    }  
    public void run()  
    {  
        b.display(a);  
        System.out.println("Back in Other Thread");  
    }  
    public static void main(String[] args)  
    {  
        new Deadlock();  
    }  
}
```

```
/*
The default/conventional behavior
of clone() is to do a shallow copy.
You can either override clone() to do
a deep copy or provide a separate method
to do a deep clone.
```

The simplest approach to deep cloning is to use Java serialization, where you serialize and deserialize the object and return the deserialized version. This will be a deep copy/clone, assuming everything in the tree is serializable. If everything is not serializable, you'll have to implement the deep cloning behavior yourself.

Assuming everything is serializable, the following should create a complete deep copy of the current class instance:

```
*/
```

```
import java.io.*;

class MyTest1 implements Serializable{
int dx,dy;

public MyTest1(int dx,int dy){
this.dx=dx;
this.dy=dy;
}
public void display(){
System.out.println("dx="+dx+" dy="+dy);
}
public void modify(){
dx++;
dy++;
}
}

class deep{
public static void main(String args[]) throws Exception{
MyTest1 test=new MyTest1(100,200);
ByteArrayOutputStream baos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(baos);
oos.writeObject(test);
```

```
ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
ObjectInputStream ois = new ObjectInputStream(bais);
MyTest1 deepCopy = (MyTest1)ois.readObject();
test.display();
deepCopy.display();
test.modify();
System.out.println("after modification of original object ");
test.display();
deepCopy.display();
}
}
```

ramireddy.satya

class A

```
{  
    int x;  
    A()  
    {  
        x=10;  
    }  
    void show()  
    {  
        System.out.println("x value is:"+x);  
    }  
}  
class B extends A  
{  
    int y;  
    B()  
    {  
        x=100;  
        y=200;  
    }  
    void show()  
    {  
        System.out.println("y value is :" +y);  
        super.show();  
    }  
}  
class Demo  
{  
    public static void main(String[] args)  
    {  
        B b1 = new B();  
        b1.show();  
    }  
}
```

```
import java.io.*;  
  
class dirdemo  
{  
    public static void main(String[] args)  
    {  
        File f = new File("/work6pm");  
        String s[]={}; //method retrieves files from  
        directory  
        for(int i=0;i<s.length;i++)  
        {  
            System.out.println(s[i]);  
        }  
    }  
}
```

a

```
import java.io.*;  
  
class disdemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileOutputStream fos = new FileOutputStream("abc1.txt");  
        DataOutputStream dos = new DataOutputStream(fos);  
        dos.writeInt(10);  
        dos.writeDouble(33.75);  
  
        FileInputStream fis = new FileInputStream("abc1.txt");  
        DataInputStream dis = new DataInputStream(fis);  
        System.out.println(dis.readInt());  
        System.out.println(dis.readDouble());  
    }  
}
```

```
class DLDemo implements Runnable
```

```

{
    DlDemo x;
    public static void main(String[] args)
    {
        DlDemo dd1 = new DlDemo();
        DlDemo dd2 = new DlDemo();
        System.out.println(dd1);
        System.out.println(dd2);

        Thread t1 = new Thread(dd1,"first");
        Thread t2 = new Thread(dd2,"second");

        dd1.x=dd2;
        System.out.println(dd1.x);
        dd2.x=dd1;
        System.out.println(dd2.x);

        t1.start();
        t2.start();

        try
        {
            t1.join();
            t2.join();
        }
        catch(Exception e)
        {
            System.exit(0);
        }
    }
    public synchronized void run()
    {
        System.out.println(Thread.currentThread().getName());
        try
        {
            System.out.println("before sleep");
            Thread.sleep(1000);
            System.out.println("after
sleep"+Thread.currentThread().getName());
        }
        catch(InterruptedException e)
        {
            System.out.println(e);
        }
        System.out.println(Thread.currentThread().getName());
        System.out.println("before method");
    }
}

```

```
        x.synmethod();
        System.out.println(Thread.currentThread().getName());
    }
    public void synmethod()
    {
        System.out.println(Thread.currentThread().getName());
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            System.out.println(e);
        }
        System.out.println("in synmethod");
    }
}
```

ramireddy.satya

```
class Emp
{
    int eno;
    String ename;
    String eadd;
    void getEmp()
```

```
{  
    eno=101;  
    ename="ram";  
    eadd="hyd";  
}  
void empDisplay()  
{  
    System.out.println("eno is :" + eno);  
    System.out.println("ename is :" + ename);  
    System.out.println("eadd is :" + eadd);  
}  
}  
class Pemp extends Emp  
{  
    String dept, desig;  
    void getPemp()  
    {  
        dept = "computer";  
        desig = "programmer";  
    }  
    void pempDisplay()  
    {  
        System.out.println("dept is :" + dept);  
        System.out.println("desig is :" + desig);  
    }  
}  
  
class EmpDemo  
{  
    public static void main(String[] args)  
    {  
        Pemp p1 = new Pemp();  
        p1.getEmp();  
        p1.getPemp();  
        p1.empDisplay();  
        p1.pempDisplay();  
    }  
}  
  
import java.awt.*;
```

```
import java.awt.event.*;
import java.util.*;
import java.io.*;

class EntryForm extends Frame implements ActionListener
{
    String sno,sname,smail;
    String tuple[];
    ArrayList tuplecollector;
    Button submit,next,showall,exit;
    TextField phoneno,name,mailid;
    Label L_phoneno,L_name,L_mailid;
    public void intializeComponents()
    {
        name=new TextField(20);
        phoneno=new TextField(10);
        mailid=new TextField(25);
        L_name=new Label("Person Name");
        L_phoneno=new Label("Phone Number");
        L_mailid=new Label("Email id");
        submit=new Button("Submit");
        next=new Button("Next");
        showall=new Button("Show");
        exit=new Button("Exit");
    }
    public void addComponents()
    {
        name.setBounds(220,50,100,20);
        add(name);
        phoneno.setBounds(220,90,100,20);
        add(phoneno);
        mailid.setBounds(220,130,110,20);
        add(mailid);
        L_name.setBounds(120,50,100,25);
        add(L_name);
        L_phoneno.setBounds(120,90,100,25);
        add(L_phoneno);
        L_mailid.setBounds(120,130,110,25);
        submit.setBounds(20,170,100,25);
        add(submit);
        next.setBounds(140,170,100,25);
        add(next);
        showall.setBounds(260,170,100,25);
        add(showall);
        exit.setBounds(380,170,100,25);
        add(exit);
    }
}
```

```

        }
    public void addComponentListeners()
    {
        submit.setActionCommand("Submit");
        submit.addActionListener(this);
        next.setActionCommand("Next");
        next.addActionListener(this);
        showall.setActionCommand("Show");
        showall.addActionListener(this);
        exit.setActionCommand("Exit");
        exit.addActionListener(this);
    }
    public void showForm()
    {
        tuple=new String[3];
        tuplecollector=new ArrayList();
        setLayout(null);
        setTitle("Mail id Entry Form");
        intializeComponents();
        addComponents();
        addComponentListeners();
        setSize(500,200);
        setVisible(true);
        setResizable(false);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getActionCommand().equals("Next"))
        {
            tuple=new String[3];
            System.out.println(name.getText()+'/'+phoneno.getText()+'/'+mailid.getText());
            tuple[0]=new String(name.getText());
            tuple[1]=new String(phoneno.getText());
            tuple[2]=new String(mailid.getText());
            tuplecollector.add(tuple);
            name.setText("");
            phoneno.setText("");
            mailid.setText("");
        }
        else
        {
            if(e.getActionCommand().equals("Submit"))
            {
                String s[]=new String[3];
                //following things Will be dispalyed on console
                for(int i=0;i<tuplecollector.size();i++)

```

```

        {
            s=(String[])tuplecollector.get(i);
            System.out.println(s);
            System.out.println(s[0]+"/"+s[1]+"/"+s[2]);
        }
    }
else
    if(e.getActionCommand().equals("Show"))
{
    //showing all records report
    System.out.println("Show");
    this.hide();
    new DisplayForm(tuplecollector,this);
}
else
    if(e.getActionCommand().equals("Exit"))
{
    System.exit(0);
}
}
}
class DisplayForm extends Frame implements ActionListener
{
    ArrayList tc;
    Font f;
    Button eform;
    EntryForm ef;
    public DisplayForm(ArrayList tc,EntryForm ef)
    {
        this.tc=tc;
        this.ef=ef;
        f=new Font("Helvitica",Font.BOLD,10);
        eform=new Button("Show Entry Form");
        eform.setBounds(400,20,100,25);
        add(eform);
        eform.setActionCommand("showEF");
        eform.addActionListener(this);
        setLayout(null);
        setTitle("Records Display Form");
        setSize(500,300);
        setVisible(true);
    }
    public void paint(Graphics g)
    {
        String s[]=new String[3];
        g.setFont(f);

```

```
g.drawString("Name", 20,50);
g.drawString("Phone Num",140,50);
g.drawString("Mail Id",300,50);
g.drawLine(20,60,400,60);
for(int i=0,x=20,y=80;i<tc.size();i++,y+=30)

{
    s=(String[])tc.get(i);
    g.drawString(s[0],x,y);
    g.drawString(s[1],x+120,y);
    g.drawString(s[2],x+250,y);
}
}

public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().equals("ShowEF"))
    {
        ef.show();
        this.dispose();
    }
}

//the main class which invokes Frame
class MailidStorage
{
    public static void main(String[] args)
    {
        EntryForm ef=new EntryForm();
        ef.showForm();
    }
}
```

```
import java.io.*;
//import java.util.Date;

class FileDemo
{
    public static void main(String[] args)
    {
        File f1 = new File("abc.txt");
        boolean b=f1.exists();
        System.out.println("file exists yes or no"+b);
        System.out.println("the length of file :" +f1.length());
        System.out.println("the name of file:" +f1.getName());
        System.out.println("the parent of file :" +f1.getParent());
        System.out.println("the path of file is :" +f1.getPath());
        System.out.println("the absolute path is :" +f1.getAbsolutePath());
        //System.out.println("the modified date is :" +(new
Date(f1.lastModified())));
        System.out.println((f1.lastModified()));

    }
}
```

```
import java.io.*;

class FileRead
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis = new FileInputStream("abc.txt");
        FileOutputStream fos = new FileOutputStream("xyz.txt");

        int k;
        while((k=fis.read())!=-1)
        {
            System.out.print((char)k);
            fos.write(k); //writing data onto disk
        }
        fos.close();
        fis.close();
    }
}
```

```
import java.io.*;
class finallydemo
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis=null;

        try
        {
            fis = new FileInputStream("abc.txt");
            int a=10;
            int b=0;
            int c=a/b;
        }
        //catch(Exception e)
        //{
        //    System.out.println(e);
        //}
        finally
        {
            System.out.println("we are in finally");

            //try
            //{
            fis.close();
            //}
            //catch(Exception e)
            //{
            //    System.out.println(e);
            //}
        }
    }
}
```

```
import java.io.*;
class finallytest
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis=null;
        try
        {
            fis= new FileInputStream("abc.txt");
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is :" +c);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        finally
        {
            System.out.println("we are in finally block");
            fis.close();
        }
    }
}
```

```
class First extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println(i);
        }
    }
    public static void main(String[] args)
    {
        First f1 = new First();
        First f2 = new First();

        f1.start();
        f2.start();
    }
}
```

```
class firstexp
{
    public static void main(String[] args)
    {
        try
        {
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is"+c);
            int d[]={10,20};
            d[5]=50;
        }
        //System.out.println("try is over and enter into catch block");
        catch(Exception e)
        {
            System.out.println(e);
            e.printStackTrace();
            System.out.println(e.getMessage());
            String s=e.toString();
            System.out.println(s);
        }
        //System.out.println("one");
        //System.out.println("two");
        //System.out.println("three");
        //System.out.println("four");
    }
}
```

```
class FirstThread extends Thread
{
    public static void main(String[] args)
    {
        FirstThread ft = new FirstThread();
        ft.start();
        FirstThread ft1 = new FirstThread();
        ft1.start();
    }
    public void run()
    {
        System.out.println("control entered into run method");

        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
}
```

```
class Fourth extends Thread
{
    public static void main(String[] args)
```

```
{  
    Fourth f1 = new Fourth();  
    System.out.println(f1);  
  
    Fourth f2 = new Fourth();  
    System.out.println(f2);  
}  
}
```

```
import java.util.*;  
  
class hashsetdemo  
{  
    public static void main(String[] args)  
    {  
        HashSet h = new HashSet();  
  
        h.add("10");  
        h.add("20");  
        h.add("30");  
        h.add(null);  
        h.add(null);  
        h.add("30");  
  
        System.out.println(h);  
    }  
}
```

```
import java.util.*;
class HashTableDemo
{
    public static void main(String[] args)
    {
        Hashtable numbers = new Hashtable();
        numbers.put("one", new Integer(1));
        numbers.put("two", new Integer(2));
        numbers.put("three", new Integer(3));
        //To retrieve a number, use the following code:
        Integer n = (Integer)numbers.get("two");
        int x= n.intValue();
        x+=5;
        numbers.put("two",new Integer(x));
        if (n != null)
            System.out.println("two = " + n);
        Enumeration e=numbers.elements();
        while(e.hasMoreElements())
        {
            System.out.println(e.nextElement()+" "+ "\n");
        }
    }
}
```

```
import java.util.*;
```

```
import java.util.*;

class hmapdemo
{
    public static void main(String []a)
    {
        //TreeMap h=new TreeMap();
        //HashMap h = new HashMap();
        //LinkedHashMap h = new LinkedHashMap();

        //Hashtable h = new Hashtable();
        Properties h = new Properties();

        Integer i1=new Integer(3);
        h.put(i1,"z");
        //h.put(null,"10");
        h.put(new Integer(2),"x");
        h.put(new Integer(1),"q");
        h.put(new Integer(4),"q");

        Set s=h.keySet();

        Iterator i=s.iterator();

        while(i.hasNext())
        {
            Object o=i.next();

            System.out.println("key="+o+" value="+h.get(o));
        }
    }
}
```

```
class httabledemo
{
    public static void main(String a[])
    {
        Hashtable ht=new Hashtable();
        ht.put(null,"activenet");
        ht.put("a",new String("java2"));
        ht.put("b",new String("c/c++"));
        ht.put( "c",new String("java"));
        ht.put("d",new String("java1"));

        Set s=ht.keySet();
        Object o1[]=s.toArray();
        for(int i=0;i<=o1.length;i++)
        {
            System.out.println("the value is"+o1[i].toString());
        }
        Iterator i=s.iterator();
        while(i.hasNext())
        {
            Object o=i.next();
            System.out.println(ht.get(o));
        }
    }
}
```

```
import java.util.*;

public class i18nsample
{
    public static void main(String args[])
    {
        String language;
        String country;
        if(args.length!=2)
        {
            language=new String("en");
            country = new String("US");
        }
        else
        {
            language = new String(args[0]);
            country = new String(args[1]);
        }
        Locale currentlocale;
        ResourceBundle messages;

        currentlocale = new Locale(language,country);

        messages = ResourceBundle.getBundle("messagesbundle",currentlocale);

        System.out.println(messages.getString("greetings"));
        System.out.println(messages.getString("inquiry"));
        System.out.println(messages.getString("farewell"));
    }
}
```

```
import java.io.*;

class inputdemo
{
    public static void main(String[] args) throws Exception
```

```
{  
    FileInputStream fis = new FileInputStream("abc.txt");  
    int x=fis.available();  
  
    System.out.println("the available bytes are"+x);  
    boolean b = fis.markSupported();  
    System.out.println("the mark is supported or not"+b);  
  
    String s="hello how r u";  
    byte by[] = s.getBytes();  
    ByteArrayInputStream bis = new ByteArrayInputStream(by);  
  
    boolean b1 = bis.markSupported();  
    System.out.println(b1);  
}  
}
```

```
class InsufficientFundsException extends Exception  
{  
    InsufficientFundsException(String s)  
    {  
        super(s);  
    }  
}
```

```
interface calculate  
{  
    void sum();  
    void sub();
```

```
    void mul();
}

class number implements calculate
{
    int a,b;
    void getdata()
    {
        a=10;
        b=20;
    }
    public void sum()
    {
        int x=a+b;
        System.out.println("the sum is :" +x);
    }
    public void sub()
    {
        int x=a-b;
        System.out.println("the sub is :" +x);
    }
    public void mul()
    {
        int x=a*b;
        System.out.println("the mul is :" +x);
    }
}
class interfacedemo
{
    public static void main(String[] args)
    {
        number n = new number();
        n.getdata();
        n.sum();
        n.sub();
        n.mul();
    }
}
```

interface A

```

{
    void one();
}

interface B
{
    void two();
}

class Demo implements A,B
{
    public void one()
    {
        System.out.println("one method is implemented from A");
    }
    public void two()
    {
        System.out.println("two method is implemented from B");
    }
}
class interfacedemo0
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();
        d.two();

        A x;
        x=d;
        x.one();
        x.two(); //this reference is not available
                  // in interface A
    }
}

```

```
interface A
{
    void one();
}

interface B
{
    void one();
}

class Demo implements A,B
{
    public void one()
    {
        System.out.println("one method is implemented");
    }
}
class interfacedemo1
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();

        A x;
        x=d;
        x.one();

        B x1;
        x1=d;
        x1.one();
    }
}

// program on same method with different return types
interface A
```

```

{
    int one();
}
interface B
{
    void one();
}

class Demo implements A,B
{
    public void one()
    {
        System.out.println("one method is implemented");
    }
    public int one()
    {
        System.out.println("one method with int return is implemented");
    }
}
class interfacedemo2
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();

        A x;
        x=d;
        x.one();

        B x1;
        x1=d;
        x1.one();
    }
}

```

```
interface A
{
    void one();
}
abstract class B
{
    void one()
    {
        System.out.println("abstract class one method implemented");
    }
}

class Demo extends B implements A
{
    public void one()
    {
        System.out.println("one method is implemented in demo class");
    }
}

class interfacedemo3
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();
    }
}
```

```
interface A
{
    void one();
}

abstract class B implements A
{
    public void one()
    {
        System.out.println("abstract class one method implemented");
    }
}

class Demo extends B
{
    public void one()
    {
        System.out.println("one method is implemented in demo class");
    }
}

class interfacedemo4
{
    public static void main(String[] args)
    {
        Demo d = new Demo();
        d.one();
    }
}
```

```
import java.io.*;

class ioredirect
{
    public static void main(String[] args)
    {
        FileInputStream fis = null;
        try
        {
            fis=new FileInputStream(args[0]);
        }
        catch(FileNotFoundException e)
        {
            System.exit(0);
        }
        try
        {
            System.setIn(fis);
            int i;
            while((i=System.in.read())!=-1)
                System.out.print((char)i);
            fis.close();
        }
        catch(Exception e)
        { }
    }
}
```

```
import java.io.*;  
  
class ioredirect2  
{  
public static void main(String[] args) throws Exception  
{  
    FileOutputStream fos = new FileOutputStream("ram.txt");  
    PrintStream pis = new PrintStream(fos);  
    System.setErr(pis);  
    int a=10;  
    int b=0;  
    int c=a/b;  
    //System.setOut(pis);  
    //System.out.print("hai hello how r u");  
}  
}
```

ramireddy.satya

```
public class Iyear  
{
```

```
String classname,staffname;
int no;
Student stu[];
public Iyear(String c,String s,int n,Student st[])
{
    classname =c;
    staffname =s;
    no=n;
    stu = new Student[no];
    for(int i=0;i<no;i++)
        stu[i]=st[i];
}
public static void main(String[] args)
{
    Iyear m;
    int mk1[]={73,84,95};
    int mk2[]={56,76,84};
    int mk3[]={73,84,95};
    int mk4[]={84,73,95};
    int mk5[]={95,73,84};

    Student st[];
    st = new Student[5];
    st[0]=new Student(2124,"hari",mk1);
    st[1]=new Student(2125,"mohan",mk2);
    st[2]=new Student(2126,"srinu",mk3);
    st[3]=new Student(2123,"prasanth",mk4);
    st[4]=new Student(2127,"ram",mk5);

    m= new Iyear("M.C.A","surya",5,st);
}
```

```
class joindemo extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args) throws Exception
    {
        joindemo j1 = new joindemo();
        joindemo j2 = new joindemo();
        //try
        //{
        j1.start();
        j1.join();

        j2.start();
        j2.join();
        //}
        //catch(Exception e)
        //{
        System.out.println("the main method is completed");
    }
}
```

```
import java.io.*;  
  
class KeyboardRead  
{  
    public static void main(String[] args) throws Exception  
    {  
        //FileInputStream fis = new FileInputStream("abc.txt");  
        BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
        String s=null;  
        System.out.println("enter text, if wants to close enter stop");  
        do  
        {  
            s=br.readLine();  
            System.out.println(s);  
        }while(!s.equals("stop"));  
    }//main  
}//class
```

```
import java.awt.*;  
  
class LabelDemo extends Frame  
{  
    LabelDemo()  
    {  
        setLayout(new FlowLayout());  
  
        Label l1 = new Label("Name");  
        Label l2 = new Label("Father name");  
  
        add(l1);  
        add(l2);  
        setSize(300,400);  
        show();  
    }  
    public static void main(String[] args)  
    {  
        new LabelDemo();  
    }  
}
```

```
import java.io.*;  
  
class LineNumberDemo  
{  
    public static void main(String[] args)  
    {  
        try  
        {  
            FileInputStream fis = new FileInputStream(args[0]);  
            LineNumberInputStream lis = new LineNumberInputStream(fis);  
            DataInputStream dis = new DataInputStream(lis);  
            String s=null;  
            do  
            {  
                s = dis.readLine();  
                System.out.println(lis.getLineNumber()+" "+s);  
            }while(!s.equals(null));  
        }  
        catch(Exception e)  
        {  
        }  
    }  
}
```

```
import java.util.*;  
  
class linkedhash  
{
```

```
public static void main(String[] args)
{
    LinkedHashSet l = new LinkedHashSet();
    l.add("10");
    l.add("20");
    l.add("30");
    l.add("40");
    l.add(null);

    System.out.println(l);
}
```

```
import java.util.*;

class LinkedListDemo
{
    public static void main(String[] args)
    {
        LinkedList l = new LinkedList();
        l.add("10");
        l.add("50");
        l.add(null);
        l.add(null);
        l.add("50");
        l.add("30");
        l.add("20");
        System.out.println(l);
    }
}
```

```
import java.awt.*;

class ListDemo extends Frame
{
    ListDemo()
    {
        setLayout(new FlowLayout());
        List l = new List();
        l.add("corejava");
        l.add("adv.java");
        l.add("j2ee");
        l.add(".net");

        add(l);
        setSize(300,400);
        show();
    }

    public static void main(String[] args)
    {
        new ListDemo();
    }
}
```

```
public class LogicalOperatorsDemo {
    public static void main(String args[]) {
```

```
boolean x = true;
boolean y = false;
System.out.println("x & y : "+(x & y));
System.out.println("x && y : "+(x && y));
System.out.println("x | y : "+(x | y));
System.out.println("x || y: "+(x || y));
System.out.println("x ^ y : "+(x ^ y));
System.out.println("!x : "+(!x));
}
}
```

```
public class MarksOutOfBoundsException extends Exception
{
```

```
    MarksOutOfBoundsException(String s)
    {
        super(s);
    }
}
```

```
import java.awt.*;
```

```
import java.awt.event.*;
class MenuTest1 extends Frame implements ActionListener
{
    MenuBar mb = new MenuBar();
    Menu m1 = new Menu("File");
    Menu m2 = new Menu("Edit");
    Menu sub = new Menu("Draw");
    MenuItem mi1 = new MenuItem("New", new MenuShortcut(KeyEvent.VK_A));
    MenuItem mi2 = new MenuItem("Open");
    MenuItem mi3 = new MenuItem("Save");
    MenuItem mi4 = new MenuItem("Save as");
    MenuItem mi5 = new MenuItem("Exit");
    MenuItem mi6 = new MenuItem("Copy");
    MenuItem smi1 = new MenuItem("Line");
    MenuItem smi2 = new MenuItem("Rect");
    TextField tf = new TextField(20);
    MenuTest1()
    {
        setSize(200,300);
        m1.add(mi1);
        m1.add(mi2);
        m1.add(mi3);
        m1.add(mi4);
        m1.addSeparator();
        m1.add(mi5);
        m1.add(mi6);
        m2.add(sub);
        sub.add(smi1);
        sub.add(smi2);
        mb.add(m1);
        mb.add(m2);
        setMenuBar(mb);
        add(tf,"North");
        mi1.addActionListener(this);
        mi2.addActionListener(this);
        mi3.addActionListener(this);
        mi4.addActionListener(this);
        mi5.addActionListener(this);
        mi6.addActionListener(this);
        smi1.addActionListener(this);
        smi2.addActionListener(this);
        setVisible(true);
    }
    public static void main(String args[])
    {
        new MenuTest1();
    }
}
```

```
    }
    public void actionPerformed(ActionEvent ae)
    {
        tf.setText("u selected"+ae.getActionCommand());
    }
}
```

ramireddy.satya

```
import java.awt.*;  
  
class MyButton extends Frame  
{  
    MyButton()  
    {  
        Button b1 = new Button("OK");  
        add(b1);  
        Button b2 = new Button("CANCEL");  
        add(b2);  
        setSize(300,400);  
        setVisible(true);  
    }  
    public static void main(String[] args)  
    {  
        new MyButton();  
    }  
}
```

```
class mycomparable implements Comparable
{
    int m1,m2;
    public mycomparable(int x,int y)
    {
        m1=x;
        m2=y;
    }
    public int compareTo(Object obj)
    {
        mycomparable t = (mycomparable)obj;
        if((this.m1==t.m1)&&(this.m2==t.m2))
            return 1;
        else
            return 0;
    }
    public static void main(String args[])
    {
        mycomparable t1 = new mycomparable(10,20);
        mycomparable t2 = new mycomparable(10,20);
        if(t1.compareTo(t2)==1)
            System.out.println("both are same");
        else
            System.out.println("both are different");
    }
}
```

```
import java.awt.*;
class MyFrame extends Frame
{
    MyFrame()
    {
        Button b1 = new Button("OK");
        add("North",b1);
        Button b2 = new Button("CANCEL");
        add("East",b2);

        Button b3 = new Button("FIRSTHELP");
        add("West",b3);
        Button b4 = new Button("CANCEL");
        add("South",b4);
        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
        new MyFrame();
    }
}
```

```
import java.awt.*;
class MyFrame1 extends Frame
{
    MyFrame1()
    {
        setLayout(new FlowLayout());

        Button b1 = new Button("OK");
        Button b2 = new Button("CANCEL");
        Button b3 = new Button("FIRSTHELP");
        Button b4 = new Button("CANCEL");
        Button b5 = new Button("FIRSTHELP");
        Button b6 = new Button("CANCEL");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);

        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
        new MyFrame1();
    }
}
```

```
import java.awt.*;
class MyFrame2 extends Frame
{
    MyFrame2()
    {
        setLayout(new GridLayout(3,4));

        Button b1 = new Button("OK");
        Button b2 = new Button("CANCEL");
        Button b3 = new Button("FIRSTHELP");
        Button b4 = new Button("CANCEL");
        Button b5 = new Button("FIRSTHELP");
        Button b6 = new Button("CANCEL");
        Button b7 = new Button("CANCEL");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        add(b7);

        setSize(100,200);
        setVisible(true);
        setTitle("first program");
    }
    public static void main(String[] args)
    {
        new MyFrame2();
    }
}
```

```
import java.awt.*;
class MyFrame3 extends Frame
```

```
{  
    MyFrame3()  
    {  
        setLayout(null);  
  
        Button b1 = new Button("OK");  
        Button b2 = new Button("CANCEL");  
        Button b3 = new Button("FIRSTHELP");  
        Button b4 = new Button("CANCEL");  
        Button b5 = new Button("FIRSTHELP");  
        Button b6 = new Button("CANCEL");  
        Button b7 = new Button("CANCEL");  
        b1.setBounds(30,50,80,90);  
        add(b1);  
  
        b2.setBounds(50,60,120,130);  
        add(b2);  
        add(b3);  
        add(b4);  
        add(b5);  
        add(b6);  
        add(b7);  
  
        setSize(100,200);  
        setVisible(true);  
        setTitle("first program");  
    }  
    public static void main(String[] args)  
    {  
        new MyFrame3();  
    }  
}
```

```
import java.util.*;
```

```
class MyStrTokenizer
{
    // "I am working in activenet as a Faculty."
    // "I am residing at SRnagar.";
    public static void main(String[] args) throws Exception
    {
        String str="My name is rami reddy.";
        StringTokenizer st=
            new StringTokenizer(str);
        int i=st.countTokens();
        System.out.println("count="+i);
        while(st.hasMoreTokens())
        {
            try
            {
                String key =st.nextToken();
                System.out.println(key);
            }
            catch(NoSuchElementException e)
            { }
        }
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
```

```
class MyWindow extends Frame implements WindowListener
{
    MyWindow()
    {
        setSize(300,400);
        show();
        addWindowListener(this);
    }
    public void windowOpened(WindowEvent we)
    { }
    public void windowActivated(WindowEvent we)
    {
        System.out.println("window activated");
    }
    public void windowDeactivated(WindowEvent we)
    { }
    public void windowIconified(WindowEvent we)
    {
        System.out.println("window is iconified");
    }
    public void windowDeiconified(WindowEvent we)
    {
        System.out.println("window deiconified");
    }
    public void windowClosed(WindowEvent we)
    { }
    public void windowClosing(WindowEvent we)
    { }

    public static void main(String[] args)
    {
        new MyWindow();
    }
}
```

```
import java.awt.*;
import java.awt.event.*;

class MyWindow1 extends Frame
{
    MyWindow1()
    {
        setSize(300,400);
        show();
        //MyAdapter ma = new MyAdapter();
        //addWindowListener(ma);
        addWindowListener(new MyAdapter());
    }

    public static void main(String[] args)
    {
        new MyWindow1();
    }
}

class MyAdapter extends WindowAdapter
{
    public void windowActivated(WindowEvent we)
    {
        System.out.println("window activated");
    }
}
```

```
class nestedtry
{
```

```
public static void main(String[] args)
{
    try
    {
        int a=args.length;
        int b=10;
        int c=b/a;
        System.out.println("c value is "+c);
        try
        {
            if(a==1)
                a=a/(a-a);
            if(a==2)
            {
                int d[]={10,20};
                d[5]=50;
            }
        }
        catch(ArithmeticException e)
        {
            System.out.println(e);
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println(e);
        }
    }
    catch(Exception e)
    {
        System.out.println("control came to outer catch block");
        System.out.println(e);
    }
}
//main
}//class
```

```
class NewThread implements Runnable
{
    public void run()
    { }

    public static void main(String[] args)
    {
        NewThread nt = new NewThread();
        Thread t = new Thread(nt,"first");
        System.out.println(t.isAlive());
        System.out.println("is the thread is daemon"+t.isDaemon());

        System.out.println("is the thread is daemon"+t.isDaemon());

        t.start();
        t.setDaemon(true);
        System.out.println(t.isAlive());

        System.out.println(t);
        System.out.println(t.getPriority());

        t.setPriority(Thread.MIN_PRIORITY+2);
        System.out.println(t.getPriority());
    }
}
```

```
class outer
{
    int a=10;
    static int b=20;
    class inner
    {
        int c=30;
        //static int d=40;
    }
}
class nonstaticinner
{
    public static void main(String[] args)
    {
        outer ou = new outer();
        System.out.println("a value is "+ou.a);
        System.out.println("b value is :" +outer.b);

        //outer.inner in = new outer().new inner();
        outer.inner in = ou.new inner();
        System.out.println("c value is :" +in.c);
    }
}
```

```
public class noti18n
```

```
{  
    public static void main(String args[])  
    {  
        System.out.println("hello");  
        System.out.println("how r u");  
        System.out.println("goodbye");  
    }  
}
```

```
package P1;  
  
class One  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Hello World!");  
    }  
}
```

```
import java.io.*;  
  
class onlydir  
{  
    public static void main(String[] args)  
    {  
        File f = new File("/work6pm");  
  
        FilenameFilter x = new onlyext(args[0]);  
        String s[] = f.list(x);  
        for(int i=0;i<s.length;i++)  
        {  
            System.out.println(s[i]);  
        }  
    }  
}  
class onlyext implements FilenameFilter  
{  
    String ext;  
    onlyext(String s)  
    {  
        ext=". "+s;  
    }  
    public boolean accept(File d,String name)  
    {  
        return name.endsWith(ext);  
    }  
}
```

```
class A  
{  
    void show()  
    {  
        System.out.println("A class method");  
    }  
}
```

```
        }
    void display()
    {
        System.out.println("class A display method");
    }
}
class B extends A
{
    void show()
    {
        System.out.println("B class method");
    }
}

class override
{
    public static void main(String[] args)
    {
        A a1 = new A();
        B b1 = new B();

        System.out.println(a1);
        System.out.println(b1);

        A x;
        x=a1;
        System.out.println(x);
        System.out.println(a1);
        x.show();

        x=b1;
        System.out.println(x);
        x.show();
    }
}
```

```

class A
{
    int a,b;
    void sum()
    {
        a=10;
        b=20;
        int c=a+b;
        System.out.println("the sum is :" +c);
    }
}
class B extends A
{
    int x,y;
    void sum(int x,int y)
    {
        this.x=x;
        this.y=y;
        int z=x+y;
        System.out.println("the sum is :" +z);
    }
}
class overridedemo
{
    public static void main(String args[])
    {
        B b1 = new B();
        b1.sum();
        b1.sum(50,60);
    }
}

```

```

import java.io.*;

class pipedemo extends Thread
{
    static PipedReader pr;

```

```
static PipedWriter pw;
pipedemo(String name)
{
    super(name);
}
public static void main(String[] args) throws IOException
{
    pipedemo pd1 = new pipedemo("src");
    pipedemo pd2 = new pipedemo("dst");
    pw = new PipedWriter();
    pr = new PipedReader(pw);
    pd1.start();
    pd2.start();

    try
    {
        Thread.sleep(2000);
    }
    catch(Exception e)
    {
    }
}
public void run()
{
    int j;
    try
    {
        if(getName().equals("src"))
        {
            for(int i=0;i<10;i++)
                pw.write(getName()+i);
        }
        else
            while((j=pr.read())!=-1)
                System.out.print((char)j);
    }
    catch(Exception e)
    {
    }
}
```

```
import java.awt.*;
import java.awt.event.*;

class popupexample extends Frame implements MouseListener
{
    PopupMenu pm;
    MenuItem mi1,mi2,mi3,mi4;
    popupexample()
    {
        pm = new PopupMenu("a sample popup.....");
        mi1= new MenuItem("one");
        mi2= new MenuItem("two");
        mi3= new MenuItem("three");
        mi4= new MenuItem("four");

        pm.add(mi1);
        pm.add(mi2);
        pm.add(mi3);
        pm.add(mi4);
        add(pm);
        addMouseListener(this);
        setVisible(true);
        setSize(200,200);
        addWindowListener(new WindowAdapter()

    }

    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }
}

public void mouseEntered(MouseEvent me)
{ }
public void mouseExited(MouseEvent me)
{ }
public void mouseClicked(MouseEvent me)
{ }
public void mouseReleased(MouseEvent me)
{ }
```

```
public void mousePressed(MouseEvent me)
{
    pm.show(me.getComponent(),me.getX(),me.getY());
}
public static void main(String args[])
{
    popupexample pp = new popupexample();
}
}
```

ramireddy.satya

```

class Q
{
    int n;
    synchronized int get()
    {
        System.out.println("got"+n);
        return n;
    }
    synchronized void put(int n)
    {
        this.n=n;
        System.out.println("put"+n);
    }
}
class producer implements Runnable
{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class consumer implements Runnable
{
    Q q;
    consumer(Q q)
    {
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.get();
        }
    }
}

```

```
        }
    }
class procon
{
    public static void main(String[] args)
    {
        Q q = new Q();
        new producer(q);
        new consumer(q);
        System.out.println("press ctrl -c to stop");
    }
}
```

ramireddy.Satya

```

class Q
{
    int n;
    boolean valueset=false;
    synchronized int get()
    {
        if(!valueset)
            try
            {
                wait();
            }
            catch(Exception e)
            {
                System.out.println("InterruptedException");
            }
        System.out.println("got"+n);
        valueset=false;
        notify();
        return n;
    }
    synchronized void put(int n)
    {
        if(valueset)
            try
            {
                wait();
            }
            catch(Exception e)
            {
                System.out.println("InterruptedException");
            }
        this.n=n;
        valueset=true;
        System.out.println("put"+n);
        notify();
    }
}
class producer implements Runnable
{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()

```

```
{  
    int i=0;  
    while(true)  
    {  
        q.put(i++);  
    }  
}  
}  
class consumer implements Runnable  
{  
    Q q;  
    consumer(Q q)  
    {  
        this.q=q;  
        new Thread(this,"consumer").start();  
    }  
    public void run()  
    {  
        int i=0;  
        while(true)  
        {  
            q.get();  
        }  
    }  
}  
class proconfixed  
{  
    public static void main(String[] args)  
    {  
        Q q = new Q();  
        new producer(q);  
        new consumer(q);  
        System.out.println("press ctrl -c to stop");  
    }  
}
```

```
import java.util.*;  
  
class PropertiesDemo  
{  
    public static void main(String[] args)  
    {  
        Properties states = new Properties();  
        String str;  
        states.put("Andhra Pradesh", "Hyderabad");  
        states.put("Tamil Nadu", "Chennai");  
        states.put("Karnataka", "Bangalore");  
        states.put("Kerala", "Trivendram");  
        Set capitals = states.keySet();  
        Iterator i = capitals.iterator();  
        while(i.hasNext())  
        {  
            str = (String)i.next();  
            System.out.println("the capital of " + str + " is  
" + states.getProperty(str));  
        }  
    }  
}
```

```
import java.util.*;  
import java.io.*;
```

```

class PropertiesDemo1
{
    public static void main(String[] args) throws Exception
    {
        FileOutputStream fos = new FileOutputStream("abc.pro");

        Properties states = new Properties();
        String str;
        states.put("Andhra Pradesh", "Hyderabad");
        states.put("Tamil Nadu", "Chennai");
        states.put("Karnataka", "Bangalore");
        states.put("Kerala", "Trivendram");
        states.store(fos, "c");

        Set capitals=states.keySet();
        Iterator i=capitals.iterator();
        while(i.hasNext())
        {
            str=(String)i.next();
            System.out.println("the capital of " +str+ " is
"+states.getProperty(str));
        }

        FileInputStream fis = new FileInputStream("abc.pro");
        states.load(fis);
        states.list(System.out);
    }
}

```

```

class A
{
    public int x;
}

```

```
protected int y;
A()
{
}
}
class B extends A
{
    void setdata()
    {
        x=10;
        y=20;
    }
    void display()
    {
        System.out.println("x value is:"+x);
        System.out.println("y value is:"+y);
    }
}
class C extends B
{
    static C c2 = new C();

    void show()
    {
        System.out.println("x value is:"+x);
        System.out.println("y value is:"+y);
    }
}
class protecteddemo
{
    public static void main(String[] args)
    {
        final C c1 = new C();
        c1.setdata();
        c1.show();
        c1.display();
    }
}
```

```
import java.io.*;

class rafdemo
{
    public static void main(String[] args) throws Exception
    {
        RandomAccessFile raf = new RandomAccessFile("abc.txt","rw");
        int k;
        //while((k=raf.read())!=-1)
        //{
        //    //k=raf.read();
        //    //System.out.print((char)k);
        //}

        raf.seek(15);
        System.out.println();

        while((k=raf.read())!=-1)
        {
            System.out.print((char)k);
        }
        raf.writeBytes("corejava class");

    }
}
```

// System.in -----> standard input stream which is associated with keyboard  
// system.out -----> standard output stream which is associated with monitor or console  
// system.err -----> standard error stream which is associated with console

```
//program to read a character from keyboard
class read
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter a value");
        int ch=System.in.read();
        System.out.println("the character is :"+(char)ch);
    }
}
```

```
// reading a line from keyboard
class read2
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter a line");
        int ch;
        do
        {
            ch=System.in.read();
            System.out.print((char)ch);
        }
        while(ch!='q');
    }
}
```

```
// reading string from the keyboard
import java.io.*;

class read3
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("enter a string");
        DataInputStream dis = new DataInputStream(System.in);
        //String s = dis.readLine();
        //System.out.println("the string is"+s);
        int k;
        do
        {
            k = dis.read();
            System.out.print((char)k);
        }
        while(k!='q');

    }
}
```

```
//reading a line from keyboard using character streams
import java.io.*;
```

```
class read4
```

```
{  
    public static void main(String[] args) throws Exception  
    {  
        System.out.println("enter a line");  
        BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
        String s = br.readLine();  
        System.out.println("The string is:"+s);  
    }  
}
```

```
// Reading a double or int from keyboard  
import java.io.*;  
  
class read5  
{  
    public static void main(String[] args) throws Exception  
    {  
        System.out.println("enter an integer!");  
        BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
        String s=br.readLine();  
        int x = Integer.parseInt(s);  
        System.out.println(x+50);  
    }  
}
```

```
import java.io.*;  
class readerstream  
{
```

```
public static void main(String[] args)
{
    FileWriter fw=null;
    FileReader fr = null;
    try
    {
        if(args.length!=2)
        {
            System.out.println("invalid arguments");
            return ;
        }
        fw = new FileWriter(args[0].trim());
        fr= new FileReader(args[1].trim());
        int k=0;
        while((k=fr.read())!=-1)
        {
            fw.write(k);
            System.out.print((char)k);
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    finally
    {
        try
        {
            fw.close();
            fr.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
import java.io.*;
```

```
class ReadExternal
```

```
{  
    public static void main(String[] args)  
    {  
        FileInputStream fis = null;  
        ObjectInputStream ois = null;  
        try  
        {  
            fis=new FileInputStream("external.dat");  
            ois=new ObjectInputStream(fis);  
            while(true)  
            {  
                try  
                {  
                    StudentExternal se =  
(StudentExternal)ois.readObject();  
                    se.print();  
                }  
                catch(Exception e)  
                {  
                    break;  
                }  
            }  
            ois.close();  
            fis.close();  
        }  
        catch(Exception ee)  
        {  
            System.out.println(ee.toString());  
        }  
    }  
}
```

```
class RelatedThread extends Thread
{
    public void run()
    {
        for(int i=0;i<50;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        RelatedThread r1 = new RelatedThread();
        RelatedThread r2 = new RelatedThread();

        r1.start();
        r2.start();
    }
}
```

```
class RelatedThread2 implements Runnable
{
    public synchronized void run()
    {
        System.out.println("thread invoked");
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        RelatedThread2 r1 = new RelatedThread2();
        RelatedThread2 r2 = new RelatedThread2();

        Thread t1 = new Thread(r1, "first");
        Thread t2 = new Thread(r1, "second");

        Thread t3 = new Thread(r2, "third");
        Thread t4 = new Thread(r2, "forth");

        t1.start();
        t2.start();
        t3.start();
        t4.start();
    }
}
```

```
public class RelationalOperatorsDemo
{
    public static void main(String args[])
    {
        int x = 10, y = 5;
        System.out.println("x > y : "+(x > y));
        System.out.println("x < y : "+(x < y));
        System.out.println("x >= y : "+(x >= y));
        System.out.println("x <= y : "+(x <= y));
        System.out.println("x == y : "+(x == y));
        System.out.println("x != y : "+(x != y));
    }
}
```

```
//:Rethrowing.java
// Demonstrating fillInStackTrace()
```

```
class Rethrowing
{
    public static void f() throws Exception
    {
        System.out.println("originating the exception in f()");
        throw new Exception("thrown from f()");
    }

    public static void g() throws Throwable
    {
        try
        {
            f();
        }
        catch(Exception e)
        {
            System.err.println("Inside g(), e.printStackTrace()");
            e.printStackTrace();
            throw e;
        }
    }

    public static void main(String[] args) throws Throwable
    {
        try
        {
            g();
        }
        catch(Exception e)
        {
            System.err.println("Caught in main, e.printStackTrace()");
            e.printStackTrace();
        }
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
```

```
class RubberLine extends Frame
{
    Point start = new Point();
    Point end = new Point();
    RubberLine()
    {
        setSize(300,300);
        addMouseListener(new MouseAdapter()

    public void mousePressed(MouseEvent e)
    {
        start.x=e.getX();
        start.y = e.getY();
    }

    public void mouseReleased(MouseEvent e)
    {
        end.x = e.getX();
        end.y = e.getY();
        repaint();
    }

    addMouseMotionListener(new MouseMotionAdapter()
    {

    public void mouseDragged(MouseEvent e)
    {
        end.x = e.getX();
        end.y = e.getY();
        repaint();
    }
});
```

```
        }

    );

setVisible(true);
}

public void paint(Graphics g)
{
    g.drawLine(start.x,start.y,end.x,end.y);
}

public static void main(String args[])
{
    new RubberLine();
}

}
```

ramireddy.satya

```
class SecondThread implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        SecondThread st = new SecondThread();
        Thread t = new Thread(st);
        t.start();
    }
}
```

```
import java.io.*;  
  
class sequencedemo  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileInputStream fis1 = new FileInputStream("abc.txt");  
        FileInputStream fis2 = new FileInputStream("xyz.txt");  
  
        SequenceInputStream sis = new SequenceInputStream(fis1,fis2);  
  
        int k;  
        while((k=sis.read())!=-1)  
        {  
            System.out.print((char)k);  
        }  
    }  
}
```

```

class student
{
    int sno;
    int m1,m2,m3;
    public student(int sno,int m1,int m2,int m3){
        this.sno=sno;
        this.m1=m1;
        this.m2=m2;
        this.m3=m3;
    }
    void modify(){
        m1++;m2++;m3++;
    }
    void display(){
        System.out.println("sno= "+sno+" m1="+m1+" m2="+m2+" m3="+m3);
    }
}

class result
{
    int sno;
    String result;
    public result(int sno,String result){
        this.sno=sno;
        this.result=result;
    }
    void modify(String ne){
        result=ne;
    }
    void display(){
        System.out.println("sno="+sno+" result="+result);
    }
}

class mytest implements Cloneable {

    student stud;
    result res;
    public mytest(int sno,int m1,int m2,int m3,String r){
        stud=new student(sno,m1,m2,m3);
        res=new result(sno,r);
    }
}

```

```
public Object clone() throws CloneNotSupportedException {  
    return super.clone();  
}  
  
}  
  
class test{  
    public static void main(String args[]) throws Exception {  
  
        mytest t=new mytest(101,67,78,89,"first");  
        mytest t1=(mytest)t.clone();  
        t.stud.display();  
        t.res.display();  
  
        t1.stud.display();  
        t1.res.display();  
  
        t.stud.modify();  
        t.res.modify("destention");  
        System.out.println("after modifictation ");  
        t.stud.display();  
        t.res.display();  
        t1.stud.display();  
        t1.res.display();  
    }  
}
```

```

abstract class shape
{
    abstract void area();
    void behavior()
    {
        System.out.println("this example show hierarchical inheritance");
    }
}

class Rect extends shape
{
    int length,breadth;
    void getRect()
    {
        length=20;
        breadth=30;

    }
    void area()
    {
        int z=length * breadth;
        System.out.println("the area of reactangle is :" +z);
    }
}

class square extends shape
{
    int a;
    void getsquare()
    {
        a=40;
    }
    void area()
    {
        int x=a * a;
        System.out.println("the area of square is :" +x);
    }
}

class shapedemo
{
    public static void main(String[] args)
    {
        //shape s = new shape();
        //s.area();

        Rect r1 = new Rect();
    }
}

```

```
r1.getRect();
r1.area();
r1.behavior();

square sq=new square();
sq.getsquare();
sq.area();
sq.behavior();

}
```

ramireddy.satyam

```
class Simple
{
    public static void main(String args[])
    {
        System.out.println("original main method");
        int x=10;
        main(x);
    }
    public static void main(int x)
    {
        System.out.println("x value is :" + x);
    }
}
```

```
class Simple
{
    void display()
    {
        System.out.println("hello");
    }
}
class simpledemo
{
    public static void main(String[] args)
    {
        Simple s[] = new Simple[5];
        s[0] = new Simple();
        s[0].display();
    }
}
```

```
class sleeptest extends Thread
{
    public void run()
    {
        try
        {
            for(int i=0;i<10;i++)
            {
                Thread.sleep(1000);
                System.out.println("i value is :"+i);
            }
        }catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void main(String[] args)
    {
        sleeptest s1 = new sleeptest();
        sleeptest s2 = new sleeptest();

        s1.start();
        s2.start();
    }
}
```

```
class sortarray
{
    public static void main(String[] args)
    {
        int a[]={50,30,20,60,45};

        int n=a.length;

        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                int temp;
                if(a[i]>a[j])
                {
                    temp=a[i];
                    a[i]=a[j];
                    a[j]=temp;
                }
            }
            for(int i=0;i<n;i++)
            {
                System.out.println(a[i]);
            }
        }
    }
}
```

```
import java.util.*;  
  
class sorting  
{  
    public static void main(String args[])  
    {  
        Arrays.sort(args);  
        for(int i=0;i<args.length;i++)  
        {  
            System.out.println(args[i]);  
        }  
    }  
}
```

```
class Speed  
{  
    public static void main(String[] args)  
    {  
        Speed objref = new Speed();  
        double s = objref.calcspeed(12.0,-3.0);  
        System.out.println("speed (km/h):"+s);  
    }  
    private double calcspeed(double distance,double time)  
    {  
        assert distance >0.0;  
        assert time >0.0 :"time is not a positive value:"+time;  
  
        double sp=distance/time;  
        assert sp >=0.0;  
        return sp;  
    }  
}
```

```
import java.util.*;
class stackdemo
{
    public static void main(String []a)
    {
        Stack s=new Stack();
        s.push(new Integer(33));
        s.push(new String("java"));
        s.push(new Float(45.67));
        s.push(new Integer(67));
        s.push(null);
        s.push(null);

        System.out.println(s);
        Object o=s.peek();

        System.out.println("PEEK"+o.toString());
        System.out.println(s);
        Object o1=s.pop();
        System.out.println("after deletion"+o1.toString());
        System.out.println(s);
    }
}
```

```
class StackTrace
{
    IllegalArgumentException ex;
    public static void main(String[] argv)
    {
        StackTrace st = new StackTrace();
        st.makeit();
        System.out.println("CONSTRUCTED BUT NOT THROWN");
        st.ex.printStackTrace();
        st.throwit();
        // MAY BE NOTREACHED - THINK ABOUT IT!
        System.out.println("CONSTRUCTED BUT NOT THROWN");
        st.ex.printStackTrace();
    }

    public void makeit() {
        ex = new IllegalArgumentException("Don't like the weather today");
    }
    public void throwit() throws IllegalArgumentException {
        throw ex;
    }
}
```

```
class AA
{
    static void classmethod()
    {
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}

class staticoverride
{
    public static void main(String args[])
    {
        AA a1 = new AA();
        BB b1 = new BB();

        AA x;
        x=a1;
        x.classmethod();
        x.instancemethod();

        x=b1;
        x.classmethod();
        x.instancemethod();
    }
}
```

```
class ABCD
{
    static void one()
    {
        System.out.println("class ABCD one method");
    }
    static void one(int x)
    {
        System.out.println("x vlaue is :" +x);
    }
}
class XYZ extends ABCD
{ }
class staticinheritdemo
{
    public static void main(String[] args)
    {
        XYZ x1 = new XYZ();
        x1.one();
        x1.one(10);
    }
}
```

1

```
class outer
{
    int a=10;
    static int b=20;

    class inner
    {
        int c=30;
        static int d=40;

        void display()
        {
            System.out.println("this is the method in inner class");
            System.out.println("the outer a is "+b);
        }
    }
}

class StaticInner
{
    public static void main(String args[])
    {
        outer ou = new outer();
        System.out.println("a value is :" +ou.a);
        System.out.println("b value is :" +outer.b);

        outer.inner in = new outer.inner();
        in.display();

        System.out.println("c value is :" +in.c);
        System.out.println("d value is :" +outer.inner.d);
    }
}
```

```
class outer
{
    int a=10;
    static int b=20;

    class inner
    {
        int c=30;
        //static int d=40;
    }
}

class StaticInner
{
    public static void main(String args[])
    {
        outer ou = new outer();
        System.out.println("a value is :" +ou.a);
        System.out.println("b value is :" +outer.b);

        outer.inner in = new outer.inner();
        in.display();

        System.out.println("c value is :" +in.c);
        System.out.println("d value is :" +outer.inner.d);
    }
}
```

```
class A  
{ }  
class demo  
{  
    static A a1 = new A();  
    public static void main(String[] args)  
    {  
        System.out.println("Hello World!");  
    }  
}
```

```

class AA
{
    static void classmethod()
    {
        System.out.println("AA class classmethod");
    }
    void instancemethod()
    {
        System.out.println("AA class instancemethod");
    }
}
class BB extends AA
{
    static void classmethod()
    {
        System.out.println("BB class classmethod");
    }
    void instancemethod()
    {
        System.out.println("BB class instancemethod");
    }
}

class staticoverride
{
    public static void main(String args[])
    {
        AA a1 = new AA();
        BB b1 = new BB();

        AA x;
        x=a1;
        x.classmethod();
        x.instancemethod();

        x=b1;
        x.classmethod();
        x.instancemethod();
    }
}

```

```
class stopdemo implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.println("i value is "+i);
            destroy();
        }
    }
    public static void main(String[] args)
    {
        stopdemo st = new stopdemo();
        Thread t = new Thread(st);
        t.start();
    }
}
```

```
class stringdemo
{
    public static void main(String[] args)
    {
        String str1 = new String();
        System.out.println(str1);

        stringdemo sd = new stringdemo();
        System.out.println(sd);
    }
}
```

```
class stringdemo1
{
    public static void main(String[] args)
    {
        //String str = new
        StringBuffer().append("hello").append("World").toString();
        String str="hello"+ "world";
        System.out.println(str);

    }
}
```

```

class StringDemo2
{
    public static void main(String[] args)
    {
        String s1 = new String("hello");
        String s2 = new String("hello");

        String s3 ="java world";
        String s4 =s3;

        String s5 = new String("happy birthday");
        String s6 = new String("Happy Birthday");

        String s9="hai";
        String s10="hai";

        String s11="hello";

// =====
        if(s1.equals("hello"))
            System.out.println("s1 equals hello");
        else
            System.out.println("s1 not equals hello");
//=====
        if(s1.equals(s2))
            System.out.println("s1 and s2 both are equal");
        else
            System.out.println("both are not equal");
//=====
        if(s1=="hello")
            System.out.println("s1 and hello both references same");
        else
            System.out.println("not same reference");
//=====
        if(s1==s2)
            System.out.println("s1 and s2 references are same");
        else
            System.out.println("s1 and s2 both references are not same");
//=====
        if(s3==s4)
            System.out.println("s3 and s4 references are same");
        else

```

```

        System.out.println("s3 and s4 references are not same");
//=====
        if(s9==s10)
            System.out.println("s9 and s10 references are same");
        else
            System.out.println("s9 and s10 references are not same");
//=====
        if(s1==s3)
            System.out.println("s1 and s3 references are same");
        else
            System.out.println("s1 and s3 references are not same");
//=====
        if(s1==s11)
            System.out.println("s1 and s11 references are same");
        else
            System.out.println("s1 and s11 references are not same");
//=====
        if(s5.equalsIgnoreCase(s6))
            System.out.println("s5 and s6 both contents are same");
        else
            System.out.println("s5 and s6 contents are different");
//=====
        String s7="god";
        String s8="good";
        System.out.println("s1 compare to s2 is :" + s1.compareTo(s2));
        System.out.println("s7 compare to s8 is :" + s7.compareTo(s8));
        System.out.println("s8 compare to s7 is :" + s8.compareTo(s7));
//=====
        System.out.println("s1 starts with :" + s1.startsWith("he"));
        System.out.println("s1 starts with :" + s1.startsWith("e", 1));
        System.out.println("s5 ends with :" + s5.endsWith("day"));
//=====
        String str = "activenetinformaticslimitednet";
        System.out.println("t located at :" + str.indexOf('t'));
        System.out.println("t located at :" + str.indexOf('t', 3));
        System.out.println("net located at :" + str.indexOf("net"));
        System.out.println("net located at :" + str.indexOf("net", 9));
        System.out.println("last i located at :" + str.lastIndexOf('i'));
        System.out.println("last i located at :" + str.lastIndexOf('i', 10));
        System.out.println("last net located at :" + str.lastIndexOf("net"));
//=====
        System.out.println("sub string from index 10 to end is :" + str.substring(10));
        System.out.println("sub string from index 0 to 9 is :" + str.substring(0, 9));
        System.out.println("result of s1.concat(s3) is " + s1.concat(s3));
        System.out.println("after concatenation:" + s1);
        System.out.println("replace 'I' with 'L' in s1 is :" + s1.replace('I', 'L'));

```

```
System.out.println("s1 to Uppercase is :" + s1.toUpperCase());
System.out.println("s6 to lowercase is :" + s1.toLowerCase());
String str1 = " activenet ";
System.out.println("str1 after trim :" + str1.trim());

boolean b = true;
char c = 's';
int i = 9;
double d = 99.99;

System.out.println("boolean value is :" + String.valueOf(b));
System.out.println("char value is :" + String.valueOf(c));
System.out.println("int value is :" + String.valueOf(i));
System.out.println("double value is :" + String.valueOf(d));

System.out.println("the length of s1 is :" + s1.length());
System.out.println("4th char at :" + s1.charAt(3));

System.out.println(s1.hashCode());
System.out.println(s2.hashCode());

}
```

```
import java.io.*;  
  
class Student implements Serializable  
{  
    int sno;  
    String sname;  
    transient double avg;  
    public void getdata(int a,String b,double c)  
    {  
        sno=a;  
        sname=b;  
        avg=c;  
    }  
    public void putdata()  
    {  
        System.out.println(sno+ " "+sname+ " "+avg);  
    }  
    public static void main(String[] args) throws Exception  
    {  
        Student st = new Student();  
        st.getdata(101,"ram",55.75);  
        FileOutputStream fos = new FileOutputStream("abc.cob");  
        ObjectOutputStream oos = new ObjectOutputStream(fos);  
        oos.writeObject(st);  
        st.putdata();  
    }  
}
```

```
import java.io.*;  
  
class Student1  
{  
    public static void main(String[] args) throws Exception  
    {  
        FileInputStream fis = new FileInputStream("abc.cob");  
        ObjectInputStream ois = new ObjectInputStream(fis);  
        Object o= ois.readObject();  
        Student st =(Student)o;  
        st.putdata();  
    }  
}
```

```
class Student implements Cloneable
{
    int sno;
    String sname;
    public Student(int x,String y)
    {
        sno=x;
        sname=y;
    }
    void display()
    {
        System.out.println("the sno is :" +sno);
        System.out.println("the sname is :" +sname);
    }
    void setname(String name)
    {
        sname=name;
    }
    public static void main(String[] args) throws Exception
    {
        Student s1 = new Student(101,"ram");
        s1.display();
        Student s2=(Student)s1.clone();
        s2.display();
        System.out.println();

        s1.setname("hari");
        s1.display();
        s2.display();

        System.out.println(s1);
        System.out.println(s2);
    }
}
```

```
class Studentdemo
{
    public static void main(String[] args) throws MarksOutOfBoundsException
    {
        Student s1 = new Student(101,"ram",99);
        Student s2 = new Student(102,"nikil",120);
        //
        try
        //{
            s1.findResult();
            s2.findResult();
        //}
        //catch(Exception e)
        //{
            //System.out.println(e);
        //
        }
        System.out.println("out of findresult");
    }
}
```

```
import java.io.*;  
  
public class StudentExternal implements Externalizable  
{  
    int sno;  
    String sname;  
    int m1,m2,m3;  
    public StudentExternal()  
    { }  
    public StudentExternal(int sno,String sname,int m1,int m2,int m3)  
    {  
        this.sno=sno;  
        this.sname=sname;  
        this.m1=m1;  
        this.m2=m2;  
        this.m3=m3;  
    }  
    public void writeExternal(ObjectOutput out)throws IOException  
    {  
        Integer isno= new Integer(sno);  
        Integer im1 = new Integer(m1);  
        Integer im2 = new Integer(m2);  
        Integer im3 = new Integer(m3);  
        //writing student no  
        out.write(isno.toString().getBytes());  
        out.write("\r\n".getBytes());  
        //writing student name  
        out.write(sname.getBytes());  
        out.write("\r\n".getBytes());  
        out.write(im1.toString().getBytes());  
        out.write("\r\n".getBytes());  
        out.write(im2.toString().getBytes());  
        out.write("\r\n".getBytes());  
        out.write(im3.toString().getBytes());  
        out.write("\r\n".getBytes());  
    }  
  
    public void readExternal(ObjectInput in) throws IOException  
    {  
        sno = Integer.parseInt(in.readLine());  
        sname = in.readLine();  
    }  
}
```

```
m1 = Integer.parseInt(in.readLine());
m2 = Integer.parseInt(in.readLine());
m3 = Integer.parseInt(in.readLine());
}
public void print()
{
    System.out.println("sno :" + sno + "\n name
:" + sname + "\nm1:" + m1 + "\nm2" + m2 + "\nm3" + m3);
}
}
```

ramireddy.Satya

```
public class subject
{
    String code;
    String title;
    int passingmin;
    int max;
    subject(String cd,String tit,int min,int total)
    {
        code =cd;
        title = tit;
        passingmin = min;
        max=total;
    }
    void displaysubject()
    {
        System.out.println("subject code is :" +code);
        System.out.println("name of the subject" +title);
        System.out.println("minimum marks for passing" +passingmin);
        System.out.println("maximum marks are:" +max);
    }
    public static void main(String[] args)
    {
        subject s = new subject("06ct01","java",35,100);
        s.displaysubject();
    }
}
```

```
class syn extends Thread
{
    public void run()
    {
        display();
    }
    synchronized void display()
    {
        for(int i=0;i<20;i++)
        {
            System.out.println("i vlaue is :" +i);
        }
    }
    public static void main(String[] args)
    {
        syn s1 = new syn();
        s1.start();
        syn s2 = new syn();
        s2.start();
    }
}
```

```
class syn1 implements Runnable
{
    public void run()
    {
        display();
    }
    public synchronized void display()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                Thread.sleep(1000);
            }
            catch(Exception e)
            {
            }
            System.out.println(Thread.currentThread().getName());
        }
    }
    public static void main(String[] args)
    {
        syn1 s = new syn1();

        Thread t1 = new Thread(s,"first");
        Thread t2 = new Thread(s,"second");

        t1.start();
        t2.start();
    }
}
```

```
class syn2 implements Runnable
{
    int x;
    public static void main(String[] args)
    {
        syn2 s = new syn2();

        Thread t1 = new Thread(s,"first");
        Thread t2 = new Thread(s,"second");

        t1.start();
        t2.start();
    }
    public void run()
    {
        int hold;
        System.out.println("entered into run method");
        System.out.println(Thread.currentThread().getName());
        for(int i=0;i<10;i++)
        {
            synchronized(this)
            {
                System.out.println("entered into syn block");
                System.out.println(Thread.currentThread().getName());
                hold=x+1;
                try
                {
                    Thread.sleep(1000);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
                x=hold;
                System.out.println(Thread.currentThread().getName());
                System.out.println("syn completed");
            }
        }
    }
}
```

```
public class TernaryOperatorsDemo {  
    public static void main(String args[]){  
        int x = 10, y = 12, z = 0;  
        z = x > y ? x : y;  
        System.out.println("z : "+z);  
    }  
}
```

```
class Test  
{  
    public static void main(String args[])  
    {  
        int a=10;  
        int b=0;  
        int c=a/b;  
        System.out.println("the c value i s:"+c);  
    }  
}
```

```
import java.io.IOException;

class testExceptions
{
    void method1() throws Throwable
    {
        throw new Throwable("Throwable Exception in method1");
    }
    void method2() throws Throwable
    {
        throw new IOException("Exception in method2");
    }
    try
    {
        method1();
    }
    catch(Throwable th)
    {
        throw th;
        throw th.fillInStackTrace();
    }
}
public static void main(String args[]) throws Throwable
{
    new testExceptions().method2();
}
```

```
import java.awt.*;

class TextDemo extends Frame
{
    TextDemo()
    {
        setLayout(new FlowLayout());
        TextField tf1 = new TextField(10);
        TextField tf2 = new TextField(30);
        tf2.setEchoChar('*');

        TextArea ta = new TextArea(10,20);

        add(tf1);
        add(tf2);
        add(ta);
        setSize(300,400);
        show();
    }
    public static void main(String[] args)
    {
        new TextDemo();
    }
}
```

```
class Third extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                Thread.sleep(1000);
            }
            catch(Exception e)
            {
            }
            System.out.println("i value is :" + i);
        }
    }
    public static void main(String[] args)
    {
        Third t1 = new Third();
        t1.start();
        Third t2 = new Third();
        t2.start();
        Third t3 = new Third();
        t3.start();
    }
}
```

this keyword uses are

---

---

1. differentiating the local variables and instance variables
2. constructor chaining(calling local constructors)
3. function chaining

```
class simple
{
    int a,b,c,d;
    simple(int x,int y)
    {
        a=x;
        b=y;
    }
    simple(int x,int y,int z)
    {
        this(x,y);
        c=z;
    }
    simple(int p,int q,int r,int s)
    {
        this(p,q,r);
        d=s;
    }
}
```

---

---

function chain

---

---

```
class simple
{
    int a,b;

    simple assign(int x,int y)
    {
        a=x;
        b=y;
        return this;
    }
}
```

```
    }
    simple display()
    {
        System.out.println("the a value is :" +a);
        System.out.println("the b value is :" +b);
        return this;
    }
}
class simpledemo
{
    public static void main(String args[])
    {
        simple s1 = new simple();
        s1.assign(10,20).display().assign(300,400).display();
    }
}
```

```
class throwsdemo
{
    public static void main(String[] args)
    {
        try
        {
            one();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    static void one()
    {
        try
        {
            two();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    static void two()
    {
        try
        {
            int a=10;
            int b=0;
            int c=a/b;
            System.out.println("c value is :" + c);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
class throwsdemo1
{
    public static void main(String[] args) throws Exception
    {
        one();
    }
    static void one()
    {
        two();
    }
    static void two() throws Exception
    {
        int a=10;
        int b=0;
        int c=a/b;
        System.out.println("c value is :" +c);
    }
}
```

```
import java.util.*;  
  
class Treeset  
{  
    public static void main(String[] args)  
    {  
        TreeSet t = new TreeSet();  
        t.add("20");  
        t.add("10");  
        t.add("30");  
        t.add(null);  
  
        System.out.println(t);  
    }  
}
```

```

import java.util.*;

class Emp
{
}

class Student
{
}

class Salary
{
}

class VectorDemo
{
    public static void main(String[] args)
    {
        Vector v = new Vector();
        System.out.println("capacity is :" + v.capacity());
        v.add("10");
        v.add("30");
        v.add("20");
        v.add("40");
        /*
        Enumeration e = v.elements();
        while(e.hasMoreElements())
        {
            Object o = e.nextElement();
            System.out.println("the object is :" + o);
        }
        Iterator i = v.iterator();
        while(i.hasNext())
        {
            Object o = i.next();
            System.out.println("the object is :" + o);
        }
        */
        ListIterator li = v.listIterator();
        while(li.hasNext())
        {
            Object o = li.next();
        }
    }
}

```

```
        System.out.println("the object is :" + o);
    }

    while(li.hasPrevious())
    {
        Object o1 = li.previous();
        System.out.println("the object is :" + o1);
    }
}
```

ramireddy.satyam

```
import java.io.*;

public class WriteExternal
{
    public static void main(String[] args)
    {
        try
        {
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            FileOutputStream fos = new FileOutputStream("external.dat");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            while(true)
            {
                System.out.println("enter the number (0 to stop) " );
                int sno = Integer.parseInt(br.readLine());
                if(sno==0)
                    break;
                System.out.print("enter the name");
                String sname=br.readLine();
                System.out.print("enter the mark1 :");
                int m1 = Integer.parseInt(br.readLine());
                System.out.print("enter the mark2 :");
                int m2 = Integer.parseInt(br.readLine());
                System.out.print("enter the mark3 :");
                int m3 = Integer.parseInt(br.readLine());

                StudentExternal st = new
StudentExternal(sno,sname,m1,m2,m3);
                oos.writeObject(st);
            }
            oos.flush();
            oos.close();
            fos.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
class Q
{
    int n;
    synchronized int get()
    {
        System.out.println("got"+n);
        return n;
    }
    synchronized void put(int n)
    {
        this.n=n;
        System.out.println("put"+n);
    }
}
class producer implements Runnable
{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class consumer implements Runnable
{
    Q q;
    consumer(Q q)
    {
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run()
    {
        int i=0;
```

```

        while(true)
        {
            q.get();
        }
    }
class procon
{
    public static void main(String[] args)
    {
        Q q = new Q();
        new producer(q);
        new consumer(q);
        System.out.println("press ctrl -c to stop");
    }
}
=====
modified version
=====
class Q
{
    int n;
    boolean valueset=false;
    synchronized int get()
    {
        if(!valueset)
            try
            {
                wait();
            }
            catch(Exception e)
            {
                System.out.println("InterruptedException");
            }
        System.out.println("got"+n);
        valueset=false;
        notify();
        return n;
    }
    synchronized void put(int n)
    {
        if(valueset)
            try
            {
                wait();
            }

```

```
        catch(Exception e)
        {
            System.out.println("InterruptedException");
        }
        this.n=n;
        valueset=true;
        System.out.println("put"+n);
        notify();
    }
}
class producer implements Runnable
{
    Q q;
    producer(Q q)
    {
        this.q=q;
        new Thread(this,"producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class consumer implements Runnable
{
    Q q;
    consumer(Q q)
    {
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.get();
        }
    }
}
class proconfixed
{
```

```

public static void main(String[] args)
{
    Q q = new Q();
    new producer(q);
    new consumer(q);
    System.out.println("press ctrl -c to stop");
}
=====
deadlockdemo
=====
class DLDemo implements Runnable
{
    DLDemo x;
    public static void main(String[] args)
    {
        DLDemo dd1 = new DLDemo();
        DLDemo dd2 = new DLDemo();
        System.out.println(dd1);
        System.out.println(dd2);

        Thread t1 = new Thread(dd1,"first");
        Thread t2 = new Thread(dd2,"second");

        dd1.x=dd2;
        System.out.println(dd1.x);
        dd2.x=dd1;
        System.out.println(dd2.x);

        t1.start();
        t2.start();

        try
        {
            t1.join();
            t2.join();
        }
        catch(Exception e)
        {
            System.exit(0);
        }
    }
    public synchronized void run()
    {
        try
        {

```

```
        System.out.println(Thread.currentThread().getName());
        Thread.sleep(1000);
    }
    catch(InterruptedException e)
    {
        System.out.println(e);
    }
    x.synmethod();
}
public void synmethod()
{
    try
    {
        Thread.sleep(1000);
    }
    catch(InterruptedException e)
    {
        System.out.println(e);
    }
    System.out.println("in synmethod");
}
}
```

```
class yieldingthread extends Thread
{
    int countdown=6;
    static int threadcount=0;
    yieldingthread()
    {
        super(""+++threadcount);
        start();
    }
    public String toString()
    {
        return "#" + getName() + ":" + countdown;
    }
    public void run()
    {
        while(true)
        {
            System.out.println(this);
            if(--countdown == 0)
                return;
            yield();
        }
    }
    public static void main(String[] args)
    {
        for(int i=0;i<5;i++)
        {
            new yieldingthread();
        }
    }
}
```

## innerclass

```
class WithInner
{
    class Inner
    {
        Inner()
        {
            System.out.println("i am inner class constructor");
        }
    }
}

public class InheritInner extends WithInner.Inner
{
    InheritInner(WithInner wi)
    {
        wi.super();
    }
    public static void main(String[] args)
    {
        WithInner wi = new WithInner();
        InheritInner ii = new InheritInner(wi);
    }
}
```

```
// Nested classes can access all members of all levels of the classes they are nested
within.
class MNA
{
    private void f()
        {
    }
    class A
    {
        private void g()
            {
            }
        public class B
        {
            void h()
            {
                g();
                f();
            }
        }
    }
}
class MultiNestingAccess {
    public static void main(String[] args) {
        MNA mna = new MNA();
        MNA.A mnaa = mna.new A();
        MNA.A.B mnaab = mnaa.new B();
        mnaab.h();
    }
} //:~
```

```
// Creating inner classes.

class A
{
    class B
    {
        private int i = 11;
        public int value()
        {
            return i;
        }
    }
    class C
    {
        private String label;

        C(String x)
        {
            label = x;
        }

        String readLabel()
        {
            return label;
        }
    }
}

// Using inner classes looks just like
// using any other class, within Parcel1:
public void ship(String dest)
{
    B b = new B();
    C c = new C(x);
    System.out.println(c.readLabel());
}

public static void main(String[] args)
{
    A a = new A();
    a.ship("India");
}
} //:~
```

```
// Creating instances of inner classes.

class A
{
    class B
    {
        private int i = 11;
        public int value()
        {
            return i;
        }
    }
    class C
    {
        private String label;

        C(String x)
        {
            label = x;
        }
        String readLabel()
        {
            return label;
        }
    }
    public static void main(String[] args)
    {
        A a = new A();
        // Must use instance of outer class
        // to create an instances of the inner class:
        A.B b = a.new B();
        A.C c = a.new C("India");
        String s=d.readLabel();
        System.out.println(s);
    }
} //:~
```

```
/** Demonstrate inner-inner class.  
 * is used to show that it can access non-local variables  
 * in the enclosing object.  
 */  
public class abc  
{  
    static String msg = "Hello";  
    public static void main(String[] av)  
    {  
        class xyz  
        {  
            public void display()  
            {  
                // print member of enclosing class  
                System.out.println(msg);  
            }  
        }  
        xyz x = new xyz();  
        x.display();  
    }  
}
```

```
// can inner classes can be overriden

class override
{
    private abc a;
    public class abc
    {
        public abc()
        {
            System.out.println("override.abc()");
        }
    }
    public override()
    {
        System.out.println("new override()");
        a=new abc();
    }
}
class overridedemo extends override
{
    public class abc
    {
        public abc()
        {
            System.out.println("overridedemo.abc()");
        }
    }
    public static void main(String[] args)
    {
        new overridedemo();
    }
}
```

```
class MyOuter2
{
    private String x="outer2";
    void doStuff()
    {
        String y="local variable";
        class MyInner
        {
            public void seeOuter()
            {
                System.out.println("Outer x is :" +x);
                System.out.println("string is :" +y);
            }
        }
        MyInner mi = new MyInner();
        mi.seeOuter();
    }
    public static void main(String[] args)
    {
        MyOuter2 my = new MyOuter2();
        my.doStuff();
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
class MyDia extends Dialog implements ActionListener
{
Frame fra;
Button b1,b2;
public MyDia(Frame f,String s)
{
    super(f,s);
    f=new MenuTest();
    fra=f;
    f.setTitle("New Frame");
    b1= new Button("OK");
    b2=new Button("Cancel");
    Panel p=new Panel();
    p.add(b1);
    p.add(b2);
    f.add(p,"South");
    f.setSize(200,200);
    f.setVisible(true);
    b2.addActionListener(this);
    fra.addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent w)
        {
            fra.setVisible(false);
        }
    });
}
public void actionPerformed(ActionEvent a)
{
    fra.setVisible(false);
}
}
public class MenuTest extends Frame implements ActionListener,ItemListener
{
CheckboxMenuItem cbi1,cbi2,cbi3;
static Frame f;
Color c;
public MenuTest()
{

```

```
MenuBar mb= newMenuBar();
setMenuBar(mb);
// is a method in Frame class
Menu m1=new Menu("File");
// creating menu option
MenuItem mi1=new MenuItem("Open",
new MenuShortcut(KeyEvent.VK_O));
//creating menu item
MenuItem mi2=new MenuItem("New",
new MenuShortcut(KeyEvent.VK_N));
MenuItem mi3=new MenuItem("Save",
new MenuShortcut(KeyEvent.VK_S));
MenuItem mi4=new MenuItem("Exit",
new MenuShortcut(KeyEvent.VK_X));

m1.add(mi1);//adding to the menu
m1.add(mi2);
m1.add(mi3);
m1.add(mi4);
mb.add(m1);//adding to the menubar

Menu m2=new Menu("Edit");

MenuItem mi5=new MenuItem("Cut");
//creating menu item
MenuItem mi6=new MenuItem("Copy");
MenuItem mi7=new MenuItem("Paste");
m2.add(mi5);//adding to the menu
m2.add(mi6);
m2.add(mi7);
mb.add(m2);//adding to the menubar
Menu m3=new Menu("Choice");
cbi1=new CheckboxMenuItem("Red");
cbi2=new CheckboxMenuItem("Green");
cbi3=new CheckboxMenuItem("Blue");
m3.add(cbi1);
m3.add(cbi2);
m3.add(cbi3);
mb.add(m3);
m1.addActionListener(this);
m2.addActionListener(this);
cbi1.addItemListener(this);
cbi2.addItemListener(this);
cbi3.addItemListener(this);

addWindowListener(new WindowAdapter()
```

```
{  
public void windowClosing(WindowEvent w)  
{  
    System.exit(0);  
}  
});  
}  
public void actionPerformed(ActionEvent a)  
{  
    FileDialog d1=new FileDialog(this,  
        "Open",FileDialog.LOAD);  
    FileDialog d2=new FileDialog(this,  
        "Save",FileDialog.SAVE);  
    if (a.getActionCommand().equals("Open"))  
    {  
        d1.show();  
    }  
    if (a.getActionCommand().equals("New"))  
    {  
        Frame f2=new Frame();  
        new MyDia(f2,"New");  
    }  
    if (a.getActionCommand().equals("Save"))  
    {  
        d2.show();  
    }  
    if (a.getActionCommand().equals("Exit"))  
    {  
        System.exit(0);  
    }  
}  
public void itemStateChanged(ItemEvent i)  
{  
    if(cbi1.getState()==true)  
    {  
        c=new Color(255,0,0);  
        repaint();  
    }  
    if(cbi2.getState())  
    {  
        c=new Color(0,255,0);  
        repaint();  
    }  
    if(cbi3.getState())  
    {  
        c=new Color(0,0,255);  
    }  
}
```

```
        repaint();
    }
}
public void paint(Graphics g)
{
f.setBackground(c);
}
public static void main(String[] args)
{
f=new MenuTest();
f.setSize(400,400);
f.setVisible(true);
}
}
```

ramireddy.satya

```
import java.awt.*;
import java.awt.event.*;
class MenuTest1 extends Frame implements ActionListener
{
    MenuBar mb = new MenuBar();
    Menu m1 = new Menu("File");
    Menu m2 = new Menu("Edit");
    Menu sub = new Menu("Draw");
    MenuItem mi1 = new MenuItem("New", new MenuShortcut(KeyEvent.VK_A));
    MenuItem mi2 = new MenuItem("Open");
    MenuItem mi3 = new MenuItem("Save");
    MenuItem mi4 = new MenuItem("Save as");
    MenuItem mi5 = new MenuItem("Exit");
    MenuItem mi6 = new MenuItem("Copy");
    MenuItem smi1 = new MenuItem("Line");
    MenuItem smi2 = new MenuItem("Rect");
    TextField tf = new TextField(20);
    MenuTest1()
    {
        setSize(200,300);
        m1.add(mi1);
        m1.add(mi2);
        m1.add(mi3);
        m1.add(mi4);
        m1.addSeparator();
        m1.add(mi5);
        m1.add(mi6);
        m2.add(sub);
        sub.add(smi1);
        sub.add(smi2);
        mb.add(m1);
        mb.add(m2);
        setMenuBar(mb);
        add(tf,"North");
        mi1.addActionListener(this);
        mi2.addActionListener(this);
        mi3.addActionListener(this);
        mi4.addActionListener(this);
        mi5.addActionListener(this);
        mi6.addActionListener(this);
```

```
        smi1.addActionListener(this);
        smi2.addActionListener(this);
        setVisible(true);
    }
public static void main(String args[])
{
    new MenuTest();
}
public void actionPerformed(ActionEvent ae)
{
    tf.setText("u selected"+ae.getActionCommand());
}
}
```

```
class MyOuter
{
    class MyInner
    {
    }
}
```

```
class outer
{
    int a=10;
    class inner
    {
        int b=20;
    }
}
class nonstaticouterdemo
{
    public static void main(String[] args)
    {
        outer ou = new outer();
        // outer.inner in = new outer().new inner();
        outer.inner in =ou.new inner();
        System.out.println("outer class variable is "+ou.a);
        System.out.println("innerclass variable is :" +in.b);
    }
}
```

```
import java.awt.*;
import java.awt.event.*;

class popupexample extends Frame implements MouseListener
{
    PopupMenu pm;
    MenuItem mi1,mi2,mi3,mi4;
    popupexample()
    {
        pm = new PopupMenu("a sample popup.....");
        mi1= new MenuItem("one");
        mi2= new MenuItem("two");
        mi3= new MenuItem("three");
        mi4= new MenuItem("four");

        pm.add(mi1);
        pm.add(mi2);
        pm.add(mi3);
        pm.add(mi4);
        add(pm);
        addMouseListener(this);
        setVisible(true);
        setSize(200,200);
        addWindowListener(new WindowAdapter()

    {

        public void windowClosing(WindowEvent we)
        {
            System.exit(0);
        }

    });

}

public void mouseEntered(MouseEvent me)
{ }

public void mouseExited(MouseEvent me)
{ }

public void mouseClicked(MouseEvent me)
```

```
{ }  
public void mouseReleased(MouseEvent me)  
{ }  
public void mousePressed(MouseEvent me)  
{  
    pm.show(me.getComponent(),me.getX(),me.getY());  
}  
public static void main(String args[])  
{  
    popupexample pp = new popupexample();  
}  
}
```

ramireddy.satya

```
import java.awt.*;
import java.awt.event.*;

class RubberLine extends Frame
{
    Point start = new Point();
    Point end = new Point();
    RubberLine()
    {
        setSize(300,300);
        addMouseListener(new MouseAdapter()
        {

            public void mousePressed(MouseEvent e)
            {
                start.x=e.getX();
                start.y = e.getY();
            }

            public void mouseReleased(MouseEvent e)
            {
                end.x = e.getX();
                end.y = e.getY();
                repaint();
            }
        });
        addMouseMotionListener(new MouseMotionAdapter()
        {

            public void mouseDragged(MouseEvent e)
            {
            }
        });
    }
}
```

```
        {
            end.x = e.getX();
            end.y = e.getY();
            repaint();
        }
    );
    setVisible(true);
}
public void paint(Graphics g)
{
    g.drawLine(start.x,start.y,end.x,end.y);
}
public static void main(String args[])
{
    new RubberLine();
}
}
```

```
class outer
{
    static int a=10;
    static class inner
    {
        static int b=20;
        int c=30;
    }
}
class staticouterdemo
{
    public static void main(String[] args)
    {
        outer ou = new outer();
        outer.inner in = new outer.inner();
        System.out.println("outer class variable is "+outer.a);
        System.out.println("innerclass variable is :" +outer.inner.b);
        System.out.println("inner class variable is :" +in.c);
    }
}
```

## Packages

```
package pack1;
class Varprotection
{
    int n=1;
    private int pri=2;
    protected int pro=3;
    public int pub =4;
    Varprotection()
    {
        System.out.println("default value is :" +n);
        System.out.println("private value is :" +pri);
        System.out.println("protected value is :" +pro);
        System.out.println("public value is :" +pub);
    }
}
```

```
package pack1;
class Samediff
{
    Samediff()
    {
        Varprotection v = new Varprotection();
        System.out.println("default value is :" +v.n);
        //System.out.println("private value is :" +v.pri);
        System.out.println("protected value is :" +v.pro);
        System.out.println("public value is :" +v.pub);
    }
}
```

```
        }  
    }
```

```
package pack1;  
class Samesub extends Varprotection  
{  
    Samesub()  
    {  
        System.out.println("default value is :" + n);  
        //System.out.println("private value is :" + pri);  
        System.out.println("protected value is :" + pro);  
        System.out.println("public value is :" + pub);  
    }  
}
```

```
package pack1;  
  
class MainTest  
{  
    public static void main(String[] args)  
    {  
        Varprotection v1 = new Varprotection();  
        Samesub s1 = new Samesub();  
        Samediff s2 = new Samediff();  
    }  
}
```

## pack2

```
package pack2;
import pack1.*;

class OtherDiff
{
    OtherDiff()
    {
        Varprotection v2 = new Varprotection();
        System.out.println("default value is :" + v2.n);
        System.out.println("private value is :" + v2.pri);
        System.out.println("protected value is :" + v2.pro);
        System.out.println("public value is :" + v2.pub);
    }
}
```

```
package pack2;
import pack1.*;

class OtherSub extends Varprotection
{
    OtherSub()
    {
        System.out.println("default value is "+n);
        System.out.println("private value is :" +pri);
        System.out.println("protected value is :" +pro);
        System.out.println("public value is :" +pub);
    }
}
```

```
package pack2;
import pack1.*;

class OtherMainTest
{
    public static void main(String[] args)
    {
        OtherSub os = new OtherSub();
        OtherDiff od = new OtherDiff();
    }
}
```

## Swing Programs

```
import java.awt.Component;
import java.awt.Container;
import javax.swingBoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class BoxLayoutDemo {
    public static void addComponentsToPane(Container pane) {
        pane.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));

        addAButton("Button 1", pane);
        addAButton("Button 2", pane);
        addAButton("Button 3", pane);
        addAButton("Long-Named Button 4", pane);
        addAButton("5", pane);
    }

    private static void addAButton(String text, Container container) {
        JButton button = new JButton(text);
        button.setAlignmentX(Component.CENTER_ALIGNMENT);
        container.add(button);
    }

    /**
     * Create the GUI and show it. For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("BoxLayoutDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Set up the content pane.
```

```
addComponentsToPane(frame.getContentPane());  
  
        //Display the window.  
        frame.pack();  
        frame.setVisible(true);  
    }  
  
public static void main(String[] args) {  
    //Schedule a job for the event-dispatching thread:  
    //creating and showing this application's GUI.  
    javax.swing.SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            createAndShowGUI();  
        }  
    });  
}
```

```
import java.awt.*;
import javax.swing.*;

public class BoxLayoutDemo1 extends JFrame
{
    public BoxLayoutDemo1()
    {
        super("BoxLayoutExample");
        Container con=getContentPane();
        con.setLayout(new BoxLayout(con, BoxLayout.Y_AXIS));
        addAButton("Button 1", con);
        addAButton("Button 2", con);
        addAButton("Button 3", con);
        addAButton("Long-Named Button 4", con);
        addAButton("5", con);
        pack();
        setVisible(true);
    }

    private static void addAButton(String text, Container container)
    {
        JButton button = new JButton(text);
        button.setAlignmentX(Component.CENTER_ALIGNMENT);
        container.add(button);
    }

    public static void main(String[] args) {
        BoxLayoutDemo1 demo=new BoxLayoutDemo1();
    }
}
```

```
import java.awt.BorderLayout;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class BoxLayoutSample {
    public static void main(String args[]) {
        JFrame verticalFrame = new JFrame("Vertical");
        verticalFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Box verticalBox = Box.createVerticalBox();
        verticalBox.add(new JLabel("Top"));
        verticalBox.add(new JTextField("Middle"));
        verticalBox.add(new JButton("Bottom"));
        verticalFrame.getContentPane().add(verticalBox, BorderLayout.CENTER);
        verticalFrame.setSize(150, 150);
        verticalFrame.setVisible(true);

        JFrame horizontalFrame = new JFrame("Horizontal");

        Box horizontalBox = Box.createHorizontalBox();
        horizontalBox.add(new JLabel("Left"));
        horizontalBox.add(new JTextField("Middle"));
        horizontalBox.add(new JButton("Right"));
        horizontalFrame.getContentPane()
            .add(horizontalBox, BorderLayout.CENTER);
        horizontalFrame.setSize(150, 150);
        horizontalFrame.setVisible(true);
    }
}
```

```
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.event.CaretEvent;
import javax.swing.event.CaretListener;
class CaretSample {
    public static void main(String args[]) {
        JFrame frame = new JFrame("Caret Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container content = frame.getContentPane();
        JTextArea textArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(textArea);
        content.add(scrollPane, BorderLayout.CENTER);
        final JTextField dot = new JTextField();
        dot.setEditable(false);
        JPanel dotPanel = new JPanel(new BorderLayout());
        dotPanel.add(new JLabel("Dot: "), BorderLayout.WEST);
        dotPanel.add(dot, BorderLayout.CENTER);
        content.add(dotPanel, BorderLayout.NORTH);

        final JTextField mark = new JTextField();
        mark.setEditable(false);
        JPanel markPanel = new JPanel(new BorderLayout());
        markPanel.add(new JLabel("Mark: "), BorderLayout.WEST);
        markPanel.add(mark, BorderLayout.CENTER);
        content.add(markPanel, BorderLayout.SOUTH);
        CaretListener listener = new CaretListener() {
            public void caretUpdate(CaretEvent caretEvent) {
                dot.setText(" " + caretEvent.getDot());
                mark.setText(" " + caretEvent.getMark());
            }
        };
        textArea.addCaretListener(listener);
    }
}
```

```
frame.setSize(250, 150);
frame.setVisible(true);
}
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class JMenuExample1 extends JFrame {

    public JMenuExample1() {
        super("DinnerMenu ");
        setSize(200, 200);
        setLocation(200, 200);

        JMenu file = new JMenu("File");
        file.add(new JMenuItem("open"));
        file.add(new JMenuItem("save"));
        file.add(new JMenuItem("save as"));
        JMenu New = new JMenu("New");
        New.add(new JMenuItem("java"));
        New.add(new JMenuItem("C"));
        New.add(new JMenuItem("C++"));
        file.add(New);
        file.addSeparator();
        JMenuBar menuBar = new JMenuBar();
        menuBar.add(file);
        setJMenuBar(menuBar);
    }

    public static void main(String[] args) {
        JFrame f = new JMenuExample1();
        f.setVisible(true);
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class myappli extends JFrame implements ActionListener
{
    JTextField b1,b2,b3;
    JButton jb1;

    myappli()
    {
        Container con = this.getContentPane();
        con.setLayout(new FlowLayout());
        JLabel l1 = new JLabel("first no");
        b1 = new JTextField(10);
        JLabel l2 = new JLabel("second no");
        b2 = new JTextField(10);
        JLabel l3 = new JLabel("value ");
        b3 = new JTextField(10);
        jb1 = new JButton("add");

        jb1.addActionListener(this);

        con.add(l1);
        con.add(b1);
        con.add(l2);
        con.add(b2);
        con.add(l3);
        con.add(b3);
        con.add(jb1);

        setVisible(true);
        setSize(300,400);
        pack();
    }
}
```

```
public void actionPerformed(ActionEvent ae)
{
    if(ae.getActionCommand().equals("add"))
    {
        String s1 = b1.getText();
        int x = Integer.parseInt(s1);

        String s2 = b2.getText();
        int y = Integer.parseInt(s2);

        int z= x+y;

        //Integer iobj = new Integer(z);
        //String s3 = iobj.toString();

        String s3 = new Integer(z).toString();
        b3.setText(s3);
    }
}

public static void main(String[] args)
{
    new myappli();
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class mycolor extends JFrame implements ItemListener
{
    JTextField jt1;
    JRadioButton b1,b2,b3;

    mycolor()
    {
        Container con = getContentPane();
        con.setLayout(new FlowLayout());
        jt1= new JTextField(10);

        ButtonGroup bg = new ButtonGroup();

        b1= new JRadioButton("RED",false);
        b2= new JRadioButton("BLUE",false);
        b3= new JRadioButton("GREEN",false);
        bg.add(b1);
        bg.add(b2);
        bg.add(b3);

        con.add(jt1);
        con.add(b1);
        con.add(b2);
        con.add(b3);
        b1.addItemListener(this);
        b2.addItemListener(this);
        b3.addItemListener(this);

        setSize(200,300);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
        }
    public void itemStateChanged(ItemEvent i)
    {
        /*
        if(b1==i.getSource())
            jt1.setBackground(Color.red);
        else
            if(b2==i.getSource())
                jt1.setBackground(Color.blue);
        else
            if(b3==i.getSource())
                jt1.setBackground(Color.green);
        */
        String s = i.getActionCommand();
        if(s.equals("RED"))
            jt1.setBackground(Color.red);
    }
}
public static void main(String[] args)
{
    new mycolor();
}
```

```
// OverlayTest.java
// A test of the OverlayLayout manager allowing experimentation.
//

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class OverlayTest extends JFrame {

    public OverlayTest() {
        super("OverlayLayout Test");
        setSize(500, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        final Container c = getContentPane();
        c.setLayout(new GridBagLayout());

        final JPanel p1 = new JPanel();
        final OverlayLayout overlay = new OverlayLayout(p1);
        p1.setLayout(overlay);

        final JButton jb1 = new JButton("B1");
        final JButton jb2 = new JButton("B2");
        final JButton jb3 = new JButton("B3");

        Dimension b1 = new Dimension(60, 50);
        Dimension b2 = new Dimension(80, 40);
        Dimension b3 = new Dimension(100, 60);

        jb1.setMinimumSize(b1);
        jb1.setMaximumSize(b1);
        jb1.setPreferredSize(b1);
        jb2.setMinimumSize(b2);
        jb2.setMaximumSize(b2);
        jb2.setPreferredSize(b2);
    }
}
```

```

jb3.setMinimumSize(b3);
jb3.setMaximumSize(b3);
jb3.setPreferredSize(b3);

SimpleReporter reporter = new SimpleReporter();
jb1.addActionListener(reporter);
jb2.addActionListener(reporter);
jb3.addActionListener(reporter);

p1.add(jb1);
p1.add(jb2);
p1.add(jb3);

JPanel p2 = new JPanel();
p2.setLayout(new GridLayout(2,6));
p2.add(new JLabel("B1 X", JLabel.CENTER));
p2.add(new JLabel("B1 Y", JLabel.CENTER));
p2.add(new JLabel("B2 X", JLabel.CENTER));
p2.add(new JLabel("B2 Y", JLabel.CENTER));
p2.add(new JLabel("B3 X", JLabel.CENTER));
p2.add(new JLabel("B3 Y", JLabel.CENTER));
p2.add(new JLabel(""));

final JTextField x1 = new JTextField("0.0", 4); // Button1 x alignment
final JTextField y1 = new JTextField("0.0", 4); // Button1 y alignment
final JTextField x2 = new JTextField("0.0", 4);
final JTextField y2 = new JTextField("0.0", 4);
final JTextField x3 = new JTextField("0.0", 4);
final JTextField y3 = new JTextField("0.0", 4);

p2.add(x1);
p2.add(y1);
p2.add(x2);
p2.add(y2);
p2.add(x3);
p2.add(y3);

GridBagConstraints constraints = new GridBagConstraints();
c.add(p1, constraints);

constraints.gridx = 1;
JButton updateButton = new JButton("Update");
updateButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        jb1.setAlignmentX(

```

```

        Float.valueOf(x1.getText().trim()).floatValue());
jb1.setAlignmentY(
        Float.valueOf(y1.getText().trim()).floatValue());
jb2.setAlignmentX(
        Float.valueOf(x2.getText().trim()).floatValue());
jb2.setAlignmentY(
        Float.valueOf(y2.getText().trim()).floatValue());
jb3.setAlignmentX(
        Float.valueOf(x3.getText().trim()).floatValue());
jb3.setAlignmentY(
        Float.valueOf(y3.getText().trim()).floatValue());

    p1.revalidate();
}
});
c.add(updateButton, constraints);

constraints.gridx = 0;
constraints.gridy = 1;
constraints.gridwidth = 2;
c.add(p2, constraints);
}

public static void main(String args[]) {
    OverlayTest ot = new OverlayTest();
    ot.setVisible(true);
}

public class SimpleReporter implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        System.out.println(ae.getActionCommand());
    }
}

public class JPanel extends JPanel {
    public void paint(Graphics g) {
        super.paint(g);
        int w = getSize().width;
        int h = getSize().height;

        g.setColor(Color.red);
        g.drawRect(0,0,w-1,h-1);
        g.drawLine(w/2,0,w/2,h);
        g.drawLine(0,h/2,w,h/2);
    }
}

```

```
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class OverlayTest1 extends JFrame {

    public OverlayTest1() {
        super("OverlayLayout Test");
        setSize(500, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container c = getContentPane();
        c.setLayout(new GridBagLayout());

        JPanel p1 = new JPanel();
        OverlayLayout overlay = new OverlayLayout(p1);
        p1.setLayout(overlay);

        JButton jb1 = new JButton("satya");
        JButton jb2 = new JButton("tech");
        JButton jb3 = new JButton("srisai");
        JButton jb4 = new JButton("arcade");

        Dimension b1= new Dimension(60, 80);
        Dimension b2 = new Dimension(80, 60);
        Dimension b3= new Dimension(100, 40);
        Dimension b4 = new Dimension(120, 20);

        /* jb1.setMinimumSize(b1);
        jb1.setMaximumSize(b1);
        jb1.setPreferredSize(b1);
        jb2.setMinimumSize(b2);
        jb2.setMaximumSize(b2);
        jb2.setPreferredSize(b2);
        jb3.setMinimumSize(b3);
        jb3.setMaximumSize(b3);
        jb3.setPreferredSize(b3);
```

```
        jb4.setMinimumSize(b4);
        jb4.setMaximumSize(b4);
        jb4.setPreferredSize(b4);
    */
    p1.add(jb1);
    p1.add(jb2);
    p1.add(jb3);
    p1.add(jb4);
    GridBagConstraints constraints = new GridBagConstraints();
    c.add(p1, constraints);
}

public static void main(String args[]) {
    OverlayTest1 ot = new OverlayTest1();
    ot.setVisible(true);
}
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="popupmenudemo.class" width=400 height=500>
</applet>
*/
public class popupmenudemo extends JApplet
{
    JPopupMenu pop = new JPopupMenu();
    public void init()
    {
        Container con = getContentPane();
        pop.add(new JMenuItem("undo"));
        pop.add(new JMenuItem("redo"));
        pop.addSeparator();

        pop.add(new JMenuItem("cut"));
        pop.add(new JMenuItem("copy"));
        pop.add(new JMenuItem("paste"));

        con.addMouseListener(new MouseAdapter()
        {
            public void mouseReleased(MouseEvent me)
            {
                showPopup(me);
            }
        });
    }
    void showPopup(MouseEvent e)
```

```
{  
    if(e.isPopupTrigger())  
        pop.show(e.getComponent(),e.getX(),e.getY());  
}  
}
```

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
/*  
<applet code="scrollpane.class" width=400 height=400>  
</applet>  
*/  
  
public class scrollpane extends JApplet  
{  
    public void init()  
    {  
        Container con = getContentPane();  
        con.setLayout(new BorderLayout());  
        JPanel jp = new JPanel();  
        jp.setLayout(new GridLayout(5,5));  
  
        for(int i=0;i<5;i++)  
            for(int j=0;j<5;j++)  
                jp.add(new JButton("button"+j));  
  
        int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;  
        int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;  
        JScrollPane js = new JScrollPane(jp,v,h);  
        con.add(js,BorderLayout.CENTER);  
    }  
}
```

```
//file: SplitPaneFrame.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class SplitPaneFrame {
    public static void main(String[] args) {
        String fileOne = args[0];
        String fileTwo = args[1];

        // create a JFrame to hold everything
        JFrame f = new JFrame("SplitPaneFrame");
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) { System.exit(0); }
        });
        f.setSize(300,300);
        f.setLocation(200, 200);

        Image leftImage = Toolkit.getDefaultToolkit().getImage(fileOne);
        Component left =
            new JScrollPane(new ImageComponent(leftImage));
        Image rightImage = Toolkit.getDefaultToolkit().getImage(fileTwo);
        Component right =
            new JScrollPane(new ImageComponent(rightImage));
        JSplitPane split =
            new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, left, right);
        split.setDividerLocation(100);
        f.getContentPane().add(split);

        f.setVisible(true);
    }
}
```

```
import javax.swing.SpringLayout;
import javax.swing.Spring;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.Container;
import java.awt.Component;

public class SpringDemo4 {
    /**
     * Create the GUI and show it. For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("SpringDemo4");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Set up the content pane.
        Container contentPane = frame.getContentPane();
        SpringLayout layout = new SpringLayout();
        contentPane.setLayout(layout);

        //Create and add the components.
        JLabel label = new JLabel("Label: ");
        JTextField textField = new JTextField("Text field", 15);
        contentPane.add(label);
        contentPane.add(textField);

        //Adjust constraints for the label so it's at (5,5).
        SpringLayout.Constraints labelCons =
            layout.getConstraints(label);
        labelCons.setX(Spring.constant(5));
        labelCons.setY(Spring.constant(5));

        //Adjust constraints for the text field so it's at
        //(<label's right edge> + 5, 5).
```

```

SpringLayout.Constraints textFieldCons =
    layout.getConstraints(textField);
textFieldCons.setX(Spring.sum(
    Spring.constant(5),
    labelCons.getConstraint(SpringLayout.EAST)));
textFieldCons.setY(Spring.constant(5));

//Adjust constraints for the content pane.
setContainerSize(contentPane, 5);

//Display the window.
frame.pack();
frame.setVisible(true);
}

public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread:
    //creating and showing this application's GUI.
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createAndShowGUI();
        }
    });
}

public static void setContainerSize(Container parent,
        int pad) {
    SpringLayout layout = (SpringLayout) parent.getLayout();
    Component[] components = parent.getComponents();
    Spring maxHeightSpring = Spring.constant(0);
    SpringLayout.Constraints pCons = layout.getConstraints(parent);

    //Set the container's right edge to the right edge
    //of its rightmost component + padding.
    Component rightmost = components[components.length - 1];
    SpringLayout.Constraints rCons =
        layout.getConstraints(rightmost);
    pCons.setConstraint(
        SpringLayout.EAST,
        Spring.sum(Spring.constant(pad),
        rCons.getConstraint(SpringLayout.EAST)));

    //Set the container's bottom edge to the bottom edge
    //of its tallest component + padding.
    for (int i = 0; i < components.length; i++) {
        SpringLayout.Constraints cons =

```

```
        layout.getConstraints(components[i]);
        maxHeightSpring = Spring.max(maxHeightSpring,
            cons.getConstraint(
                SpringLayout.SOUTH));
    }
    pCons.setConstraint(
        SpringLayout.SOUTH,
        Spring.sum(Spring.constant(pad),
        maxHeightSpring));
}
}
```

ramireddy.satya

```
import java.awt.*;
import javax.swing.*;

class TabbedPaneExample extends JFrame
{
    private JTabbedPane tabbedPane;
    private JPanel panel1;
    private JPanel panel2;
    private JPanel panel3;

    public TabbedPaneExample()
    {
        setTitle( "JTabbedPane Example" );
        setSize( 350, 300 );
        setBackground( Color.pink );
        JPanel topPanel = new JPanel();
        topPanel.setLayout( new BorderLayout() );
        getContentPane().add( topPanel );

        // Create the tab pages
        createPage1();
        createPage2();
        createPage3();

        // Create a tabbed pane
        tabbedPane = new JTabbedPane();
        tabbedPane.addTab( "Page 1", panel1 );
        tabbedPane.addTab( "Page 2", panel2 );
        tabbedPane.addTab( "Page 3", panel3 );
        topPanel.add( tabbedPane, BorderLayout.CENTER );
    }

    public void createPage1()
    {
        panel1 = new JPanel();
        panel1.setLayout( null );
    }
}
```

```

JLabel label1 = new JLabel( "Username:" );
label1.setBounds( 10, 15, 150, 20 );
panel1.add( label1 );

JTextField field = new JTextField();
field.setBounds( 10, 35, 150, 20 );
panel1.add( field );
JLabel label2 = new JLabel( "Password:" );
label2.setBounds( 10, 60, 150, 20 );
panel1.add( label2 );
JPasswordField fieldPass = new JPasswordField();
fieldPass.setBounds( 10, 80, 150, 20 );
panel1.add( fieldPass );
panel1.setBackground(Color.red);
}

public void createPage2()
{
    panel2 = new JPanel();
    panel2.setLayout( new BorderLayout() );
    panel2.add( new JButton( "North" ), BorderLayout.NORTH );
    panel2.add( new JButton( "South" ), BorderLayout.SOUTH );
    panel2.add( new JButton( "East" ), BorderLayout.EAST );
    panel2.add( new JButton( "West" ), BorderLayout.WEST );
    panel2.add( new JButton( "Center" ), BorderLayout.CENTER );
    //panel2.setBackground(Color.green);
}

public void createPage3()
{
    panel3 = new JPanel();
    panel3.setLayout( new FlowLayout() );
    panel3.add( new JLabel( "Field 1:" ) );
    panel3.add( new JTextField("vipula technogies",30) );
    panel3.add( new JLabel( "Field 2:" ) );
    panel3.add( new JTextField("ameerpet",30) );
    panel3.add( new JLabel( "Field 3:" ) );
    panel3.add( new JTextField("hyderabad",30) );
    panel3.setBackground(Color.blue);
}

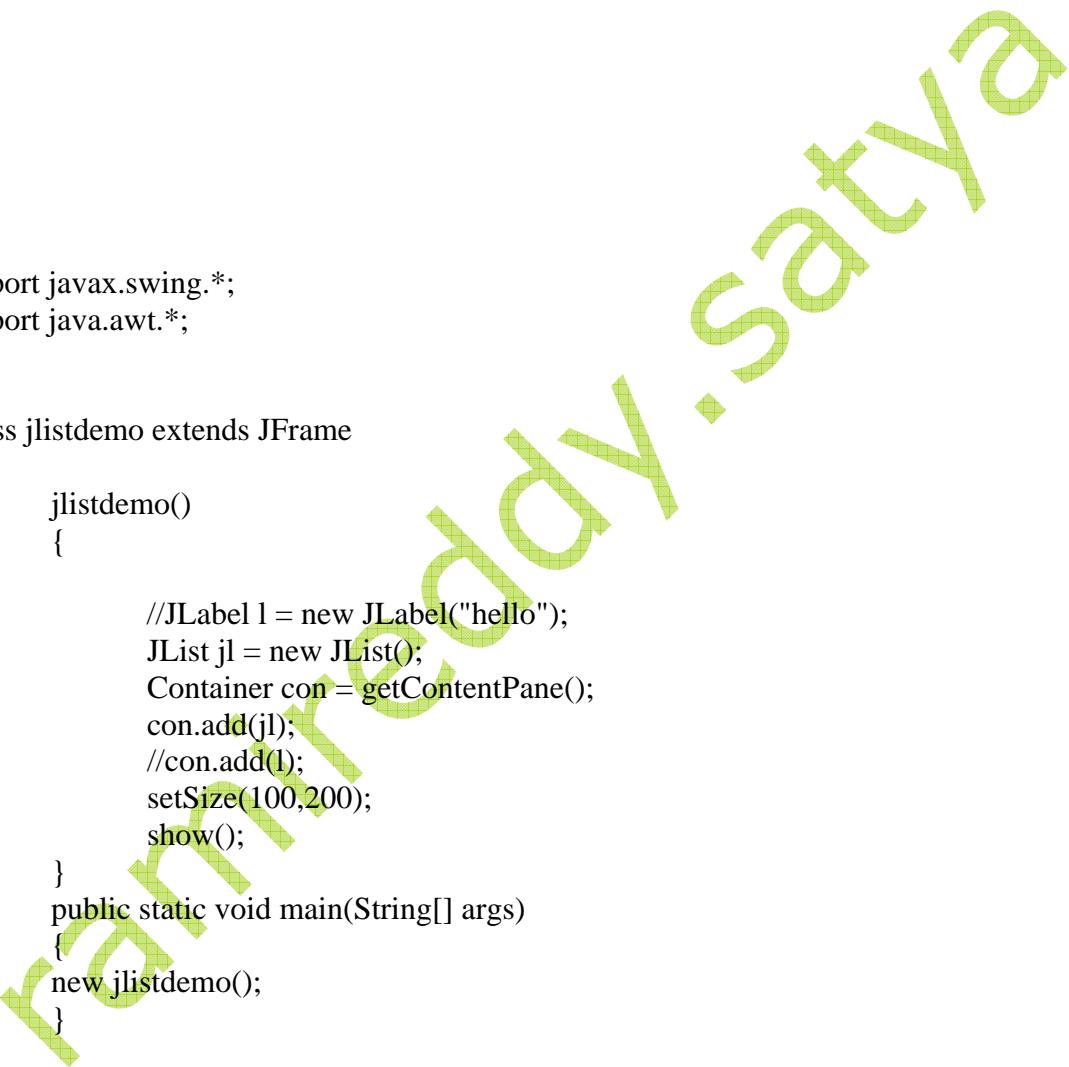
public static void main( String args[] )
{
    TabbedPaneExample mainFrame = new TabbedPaneExample();
    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainFrame.setVisible( true );
}

```

```
}
```

```
/*
<applet code=TJTable width=500 height=600>
</applet>
*/
import javax.swing.*;
import java.awt.*;
import java.util.*;

public class TJTable extends JApplet
{
    String[] colnames= {"name","size(Bytes)","Date","Directory"};
    Object[][] rowdata = {
        {"autoexec.bat","149","09-11-07",new Boolean(false)},
        {"real","dir","12-11-07",new Boolean(true)},
    };
    public void init()
    {
        JTable table = new JTable(rowdata,colnames);
        getContentPane().add(table.getTableHeader(),BorderLayout.NORTH);
        getContentPane().add(table);
    }
}
```



```
import javax.swing.*;
import java.awt.*;

class jlistdemo extends JFrame
{
    jlistdemo()
    {
        //JLabel l = new JLabel("hello");
        JList jl = new JList();
        Container con = getContentPane();
        con.add(jl);
        //con.add(l);
        setSize(100,200);
        show();
    }
    public static void main(String[] args)
    {
        new jlistdemo();
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class swing1 extends JFrame
{
    public static void main(String args[])
    {
        swing1 f=new swing1();
        f.setSize(200,200);
        f.setVisible(true);
        Container con=f.getContentPane();
        con.setBackground(Color.orange);
        //Color c1=new Color(240,245,255);
        //con.setBackground(c1);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
        con.setLayout(new FlowLayout());
        JButton b1 = new JButton("ok");
        con.add(b1);
        JButton b2 = new JButton("ok");
        con.add(b2);
    }
}
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class swing2 extends JFrame
{
    public static void main(String args[])
    {
        swing2 f=new swing2();
        f.setSize(500,300);
        f.setVisible(true);

        /*f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    public swing2()
    {
        Icon i1=new ImageIcon("paste.gif");
        Icon i2=new ImageIcon("web.gif");
        JButton b1=new JButton();
        JButton b2=new JButton("second");
        JButton b3=new JButton(i1);
        JButton b4=new JButton("fourth",i2);

        Container con=getContentPane();
        FlowLayout fl=new FlowLayout(FlowLayout.LEFT);
        //FlowLayout fl=new FlowLayout(FlowLayout.RIGHT);
        //con.setLayout(fl);
        con.setLayout(new FlowLayout(FlowLayout.RIGHT));
        con.add(b1);
        con.add(b2);
        con.add(b3);
        con.add(b4);
        b1.setText("first");
        System.out.println(b2.getText());
        b1.setIcon(b4.getIcon());
        b1.setBackground(Color.yellow);
```

```
        b1.setForeground(Color.red);
    }
}
```

```
import javax.swing.*;
import java.awt.*;

class Swing2a extends JFrame
{
    Swing2a()
    {
        Container con = getContentPane();
        con.setLayout(new FlowLayout());
        JButton b1 = new JButton("ok");
        con.add(b1);

        JButton b2 = new JButton("cancel");
        con.add(b2);
        setSize(300,400);
        setVisible(true);
    }
    public static void main(String[] args)
    {
        new Swing2a();
    }
}
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swing3 extends JFrame
{
    public static void main(String args[])
    {
        swing3 f=new swing3();
        f.setSize(500,300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing3()
    {
        JButton b1=new JButton("one");
        JButton b2=new JButton("two");
        JButton b3=new JButton("three");
        JButton b4=new JButton("four");
        JButton b5=new JButton("five");
        Container con=this.getContentPane();
        con.add("North",b1);
        con.add(BorderLayout.SOUTH,b2);
        con.add(b3, BorderLayout.EAST);
        con.add("West",b4);
        con.add(b5);
    }
}
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swing4 extends JFrame
{
    public static void main(String args[])
    {
        swing4 f=new swing4();
        f.setSize(600,400);
        f.setVisible(true);
        swing4 f1=new swing4();
        f1.setSize(600,400);
        f1.setVisible(true);

        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing4()
    {
        JButton b1=new JButton("red");
        JButton b2=new JButton("blue");
        JButton b3=new JButton("green");
        JPanel p1=new JPanel();
        p1.add(b1);p1.add(b2);p1.add(b3);
        //p1.setBackground(Color.pink);

        JButton b4=new JButton("red");
        JButton b5=new JButton("blue");
        JButton b6=new JButton("green");
        JPanel p2=new JPanel();
        p2.add(b4);p2.add(b5);p2.add(b6);
        //p2.setBackground(Color.red);

        Container con=getContentPane();
        con.add("South",p1);
        con.add("North",p2);
    }
}
```

```
    }  
}
```

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
public class swing5 extends JFrame  
{  
    public static void main(String args[])  
    {  
        swing5 f=new swing5();  
        f.setSize(700,500);  
        f.setVisible(true);  
  
        f.addWindowListener(new WindowAdapter()  
        {  
            public void windowClosing(WindowEvent w)  
            {  
                System.exit(0);  
            }  
        });  
    }  
    public swing5()  
    {  
        Icon i1=new ImageIcon("alice.gif");  
        JButton b1=new JButton("welcome",i1);  
        //b1.setForeground(Color.magenta);  
        b1.setHorizontalTextPosition(SwingConstants.LEFT);  
        b1.setVerticalTextPosition(SwingConstants.TOP);  
  
        Container con=getContentPane();  
        con.setLayout(new FlowLayout());  
        con.add(b1);  
  
        Dimension d1=b1.getPreferredSize();  
        System.out.println(d1.width+" "+d1.height);  
  
        b1.setPreferredSize( new Dimension(300,300));
```

```
        }
    }

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing6 extends JFrame
{
    public static void main(String args[])
    {
        swing6 f=new swing6();
        f.setSize(700,500);
        f.setVisible(true);

        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing6()
    {
        Icon i1=new ImageIcon("save.gif");

        JLabel l1=new JLabel("first label");
        JLabel l2=new JLabel(i1);
        JLabel l3=new JLabel("third label",JLabel.CENTER);
        l3.setIcon(i1);
        l1.setForeground(Color.red);
        l3.setForeground(Color.blue);
        l1.setBackground(Color.cyan);
        l3.setBackground(Color.cyan);
        l1.setOpaque(false);
        l3.setOpaque(true);
        l1.setPreferredSize(new Dimension(200,40));
        l3.setPreferredSize(new Dimension(200,40));
        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(l1);con.add(l2);con.add(l3);
    }
}
```

```
        }  
    }  
  
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.border.*;  
public class swing7 extends JFrame  
{  
    public static void main(String args[])  
    {  
        swing7 f=new swing7();  
        f.setSize(700,500);  
        f.setVisible(true);  
  
        f.addWindowListener(new WindowAdapter()  
        {  
            public void windowClosing(WindowEvent w)  
            {  
                System.exit(0);  
            }  
        });  
    }  
    public swing7()  
    {  
        Container con=getContentPane();  
        con.setLayout(new FlowLayout());  
  
        JButton b1=new JButton("demo button");  
        LineBorder r1=new LineBorder(Color.pink);  
        b1.setBorder(r1);  
        con.add(b1);  
  
        JButton b2=new JButton("demo button");  
        LineBorder r2=new LineBorder(Color.pink ,3);  
        b2.setBorder(r2);  
        con.add(b2);  
  
        JButton b3=new JButton("demo button");  
        TitledBorder r3=new TitledBorder(r1,"advjava");  
    }  
}
```

```
b3.setBorder(r3);
con.add(b3);

JButton b4=new JButton("demo button");
EmptyBorder r4=new EmptyBorder(10,10,10,10);
b4.setBorder(r4);
con.add(b4);

JButton b5=new JButton("demo button");
MatteBorder r5=new MatteBorder(10,10,10,10,Color.cyan);
b5.setBorder(r5);
con.add(b5);

JButton b6=new JButton("demo button");
MatteBorder r6=new MatteBorder(10,10,10,10,new ImageIcon("bullet.gif"));
b6.setBorder(r6);
con.add(b6);

JButton b7=new JButton("demo button");
EtchedBorder r7=new EtchedBorder(EtchedBorder.LOWERED);
b7.setBorder(r7);
con.add(b7);
JButton b8=new JButton("demo button");
EtchedBorder r8=new EtchedBorder(EtchedBorder.RAISED);
b8.setBorder(r8);
con.add(b8);
JButton b9=new JButton("demo button");
EtchedBorder r9=new
EtchedBorder(EtchedBorder.LOWERED,Color.red,Color.yellow);
b9.setBorder(r9);
con.add(b9);

JButton b10=new JButton("demo button");
BevelBorder r10=new BevelBorder(BevelBorder.LOWERED);
b10.setBorder(r10);
con.add(b10);
JButton b11=new JButton("demo button");
BevelBorder r11=new BevelBorder(BevelBorder.RAISED);
b11.setBorder(r11);
con.add(b11);
JButton b12=new JButton("demo button");
BevelBorder r12=new
BevelBorder(BevelBorder.LOWERED,Color.red,Color.yellow);
b12.setBorder(r12);
con.add(b12);
JButton b13=new JButton("demo button");
```

```
BevelBorder r13=new  
BevelBorder(BevelBorder.LOWERED,Color.red,Color.yellow,Color.blue,Color.cyan);  
    b13.setBorder(r13);  
    con.add(b13);  
  
    JButton b14=new JButton("demo button");  
    SoftBevelBorder r14=new  
    SoftBevelBorder(SoftBevelBorder.LOWERED,Color.red,Color.yellow,Color.blue,Color.  
cyan);  
        b14.setBorder(r14);  
        con.add(b14);  
  
    JButton b15=new JButton("demo button");  
    CompoundBorder r15=new CompoundBorder(r1,r4);  
    b15.setBorder(r15);  
    con.add(b15);  
}  
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing8 extends JFrame
{
    public static void main(String args[])
    {
        System.out.println("execution of main method..");
        swing8 f=new swing8();
        f.setSize(800,600);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public void paint(Graphics g)
    {
        System.out.println("execution of paint method..");
        g.setColor(Color.red);
        g.drawLine(30,30,150,30);
        g.drawRect(30,50,100,50);
        g.fillRoundRect(150,50,100,50,20,20);
        g.fill3DRect(30,130,100,50,true);
        g.fill3DRect(150,130,100,50,false);
        g.setColor(Color.magenta);
        g.drawOval(30,220,100,100);
        g.fillArc(30,350,100,100,90,90);
        Font f1=new Font("Arial",Font.BOLD+Font.ITALIC,30);
        g.setFont(f1);
        g.drawString("java by rami reddy",300,50);
        int x[]={500,600,550,450,400};
        int y[]={200,250,300,300,250};
        g.setColor(Color.blue);
        //g.drawPolyline(x,y,3);
        //g.drawPolygon(x,y,3);
        g.fillPolygon(x,y,5);
```

```
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
/*
<applet code="swing8a" width=400 height=200>
</applet>
*/
public class swing8a extends JApplet
{
    public void paint(Graphics g)
    {
        //System.out.println("execution of paint method..");
        g.setColor(Color.red);
        g.drawLine(30,30,150,30);
        g.drawRect(30,50,100,50);
        g.fillRoundRect(150,50,100,50,20,20);
        g.fill3DRect(30,130,100,50,true);
        g.fill3DRect(150,130,100,50,false);
        g.setColor(Color.magenta);
        g.drawOval(30,220,100,100);
        g.fillArc(30,350,100,100,90,90);
        Font f1=new Font("Arial",Font.BOLD+Font.ITALIC,90);
        g.setFont(f1);
        g.drawString("core java by rami reddy ",300,50);
        int x[]={500,600,550,450,400};
        int y[]={200,250,300,300,250};
        g.setColor(Color.blue);
        //g.drawPolyline(x,y,3);
        //g.drawPolygon(x,y,3);
        g.fillPolygon(x,y,5);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.applet.*;

/*
<applet code="swing9.class" width=200 height=200>
</applet>
*/
public class swing9 extends Applet
{
    public void init()
    {
        Button b1 = new Button("ok");
        add(b1);
        System.out.println("init method..");
    }
    public void stop()
    {
        System.out.println("stop method..");
    }
    public void start()
    {
        //Button b1 = new Button();
        //add(b1);
        System.out.println("start method..");
    }
    public void destroy()
    {
        System.out.println("destroy method..");
    }
    public void paint(Graphics g)
    {
        System.out.println("paint method..");
        g.drawString("core java by rami reddy",70,80);
    }
}
```

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/*
<applet code="swing10" width=300 height=200>
</applet>
*/

public class swing10 extends JApplet implements ActionListener
{
    JButton b1,b2;
    public void init()
    {
        b1=new JButton("first button");
        b2=new JButton("second button");

        Container con=getContentPane();
        FlowLayout fl=new FlowLayout();
        con.setLayout(fl);
        con.add(b1);con.add(b2);

        b1.addActionListener(this);
        b2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {
        JButton b=(JButton)a.getSource();
        if(b==b1)
            System.out.println("first button is clicked..");
        else if(b==b2)
            System.out.println("second button is clicked...");

        /*String str=a.getActionCommand();
        if(str.equals("first button"))
            System.out.println("first button is clicked..");
        else if(str.equals("second button"))
            System.out.println("second button is clicked..");
        */
    }
}

```

```
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/*
<applet code="swing11" width=300 height=200>
</applet>
*/
public class swing11 extends JApplet implements FocusListener,ActionListener
{
    JButton b1,b2,b3,b4;
    Container con;
    public void init()
    {
        b1=new JButton("first");
        b2=new JButton("second");
        b3=new JButton("third");
        b4=new JButton("fourth");

        con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);con.add(b2);con.add(b3);con.add(b4);

        b1.addFocusListener(this);
        b2.addFocusListener(this);
        b3.addFocusListener(this);
        b4.addFocusListener(this);
        b2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {
        int r=(int)(Math.random() * 255);
        int g=(int)(Math.random() * 255);

```

```
        int b=(int)(Math.random() * 255);
        Color c1=new Color(r,g,b);
        con.setBackground(c1);
    }
    public void focusGained(FocusEvent f)
    {
        JButton b=(JButton)f.getSource();
        b.setBackground(Color.yellow);
        b.setForeground(Color.red);
    }
    public void focusLost(FocusEvent f)
    {
        JButton b=(JButton)f.getSource();
        b.setBackground(Color.red);
        //b.setForeground(Color.black);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
/*
<applet code="swing12" width=300 height=200>
</applet>
*/
public class swing12 extends JApplet
{
    public void init()
    {
        JTextField tf1=new JTextField();
        JTextField tf2=new JTextField(30);
        JTextField tf3=new JTextField("welcome");
        JTextField tf4=new JTextField("welcome",20);
        JPasswordField tf5=new JPasswordField(20);
        tf5.setEchoChar('&');

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(tf1);con.add(tf2);con.add(tf3);
        con.add(tf4);con.add(tf5);
    }
}
```

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class swing13 extends JFrame implements ActionListener
{
    public static void main(String args[])
    {
        swing13 f=new swing13();
        f.setSize(250,200);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    //JPasswordField tf1;
    JTextField tf1,tf2,tf3;
    public swing13()
    {
        tf1=new JTextField();
        tf2=new JTextField();
        tf3=new JTextField();
        tf1.setBounds(25,30,200,30);
        tf2.setBounds(25,80,200,30);
        tf3.setBounds(25,130,200,30);
        Container con=getContentPane();
        con.setLayout(null);
        con.add(tf1);con.add(tf2);con.add(tf3);

        tf2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {

```

```
String s1=tf1.getText();
String s2=tf2.getText();
int n1=Integer.parseInt(s1);
int n2=Integer.parseInt(s2);
int n3=n1 +n2;
String s3=String.valueOf(n3);
tf3.setText(s3);

// tf3.setText(String.valueOf( Integer.parseInt(tf1.getText()) +
Integer.parseInt(tf2.getText())));
}
}
```

ramireddy.satyam

```
import javax.swing.*;
import java.awt.*;
/*
<applet code="swing14.class" width=200 height=300>
</applet>
*/
public class swing14 extends JApplet
{
    public void init()
    {
        JButton b[]=new JButton[13];
        Container con=getContentPane();
        //GridLayout gl=new GridLayout(3,4);
        GridLayout gl=new GridLayout(3,4,20,10);
        con.setLayout(gl);
        for(int i=0;i<b.length;i++)
        {
            b[i]=new JButton("button"+i);
            con.add(b[i]);
        }
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/*
<applet code="swing15.class" width=200 height=300>
</applet>
*/
public class swing15 extends JApplet implements MouseListener, MouseMotionListener
{
    public void init()
    {
        addMouseListener(this);
        addMouseMotionListener(this);
    }
    public void mouseEntered(MouseEvent m)
    {
        System.out.println(" mouse entered at :" + m.getX() + "," + m.getY());
    }
    public void mouseExited(MouseEvent m)
    {
        System.out.println(" mouse exited at :" + m.getX() + "," + m.getY());
    }
    public void mousePressed(MouseEvent m)
    {
        System.out.println(" mouse pressed at :" + m.getX() + "," + m.getY());
    }
    public void mouseReleased(MouseEvent m)
    {
        System.out.println(" mouse released at :" + m.getX() + "," + m.getY());
    }
    public void mouseClicked(MouseEvent m)
    {
        System.out.println(" mouse clicked at :" + m.getX() + "," + m.getY());
    }
    public void mouseMoved(MouseEvent m)
    {
        System.out.println(" mouse moved at :" + m.getX() + "," + m.getY());
    }
    public void mouseDragged(MouseEvent m)
    {
    }
}
```

```
        System.out.println(" mouse dragged at :" + m.getX() + "," + m.getY());
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class swing16 extends JFrame implements MouseListener
{
    public static void main(String args[])
    {
        swing16 f=new swing16();
        f.setSize(500,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing16()
    {
        addMouseListener(this);
    }
    public void mouseClicked(MouseEvent m)
    {
        Graphics g=getGraphics();
        g.setColor(Color.magenta);
        g.drawString("rami reddy",m.getX(),m.getY());
    }
    public void mouseReleased(MouseEvent m){ }
    public void mousePressed(MouseEvent m){ }
    public void mouseExited(MouseEvent m){ }
    public void mouseEntered(MouseEvent m){ }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing16a extends JFrame
{
    public static void main(String args[])
    {
        swing16a f=new swing16a();
        f.setSize(500,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing16a()
    {
        addMouseListener(new mymouse());
    }
    public class mymouse extends MouseAdapter
    {
        public void mouseClicked(MouseEvent m)
        {
            Graphics g=getGraphics();
            g.setColor(Color.magenta);
            g.drawString("rami reddy",m.getX(),m.getY());
        }
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/*
<applet code=swing17.class width=200 height=300>
</applet>
*/

public class swing17 extends JApplet implements ActionListener
{
    JButton b1,b2,b3;
    public void init()
    {
        Icon i1=new ImageIcon("cut.gif");
        Icon i2=new ImageIcon("copy.gif");
        Icon i3=new ImageIcon("paste.gif");
        Icon i4=new ImageIcon("save.gif");

        b1=new JButton("disable"); b1.setEnabled(false);
        b2=new JButton("enable");
        b3=new JButton("demobutton",i4);

        b3.setRolloverIcon(i3);
        b3.setPressedIcon(i2);
        b3.setDisabledIcon(i1);

        b1.setToolTipText("enable button");
        b2.setToolTipText("disable button");

        b1.setMnemonic('e');
        b2.setMnemonic('d');

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);con.add(b2);con.add(b3);

        b1.addActionListener(this);
    }
}
```

```
        b2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {
        JButton b=(JButton)a.getSource();
        if(b==b1)
        {
            b3.setEnabled(true);
            b1.setEnabled(false);
            b2.setEnabled(true);
        }
        else if(b==b2)
        {
            b3.setEnabled(false);
            b2.setEnabled(false);
            b1.setEnabled(true);
        }
    }
}
```

```
import javax.swing.*;
import java.awt.*;

/*
<applet code="swing18" width=300 height=300>
</applet>
*/
public class swing18 extends JApplet
{
    public void init()
    {
        JRadioButton r1=new JRadioButton("radio1");
        JRadioButton r2=new JRadioButton("radio2");
        JRadioButton r3=new JRadioButton("radio3");

        JCheckBox c1=new JCheckBox("check1");
        JCheckBox c2=new JCheckBox("check2");
        JCheckBox c3=new JCheckBox("check3");

        JToggleButton t1=new JToggleButton("toggle1");
        JToggleButton t2=new JToggleButton("toggle2");
        JToggleButton t3=new JToggleButton("toggle3");

        Box h1=Box.createHorizontalBox();
        h1.add(r1);h1.add(r2);h1.add(r3);
        Box h2=Box.createHorizontalBox();
        h2.add(c1);h2.add(c2);h2.add(c3);
        Box h3=Box.createHorizontalBox();
        h3.add(t1);h3.add(t2);h3.add(t3);

        Box v1=Box.createVerticalBox();
        v1.add(h1);v1.add(h2);v1.add(h3);

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(v1);
        ButtonGroup bg1=new ButtonGroup();
        bg1.add(r1);bg1.add(r2);bg1.add(r3);
        //ButtonGroup bg2=new ButtonGroup();
        //bg2.add(c1);bg2.add(c2);bg2.add(c3);
        //ButtonGroup bg3=new ButtonGroup();
```

```
//bg3.add(t1);bg3.add(t2);bg3.add(t3);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
/*
<applet code="swing19" width=300 height=300>
</applet>
*/
public class swing19 extends JApplet
{
    public void init()
    {
        Icon i1=new ImageIcon("copy.gif");
        Icon i2=new ImageIcon("paste.gif");
        JCheckBox c1=new JCheckBox();
        JCheckBox c2=new JCheckBox("bold");
        JCheckBox c3=new JCheckBox("bold",true);
        JCheckBox c4=new JCheckBox(i1);
        JCheckBox c5=new JCheckBox(i1,true);
        JCheckBox c6=new JCheckBox("bold",i1);
        JCheckBox c7=new JCheckBox("bold",i1,true);

        c5.setSelectedIcon(i2);
        c6.setSelectedIcon(i2);
        c7.setSelectedIcon(i2);

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(c1);con.add(c2);con.add(c3);
        con.add(c4);con.add(c5);con.add(c6);con.add(c7);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class swing20 extends JFrame implements ItemListener
{
    public static void main(String args[])
    {
        swing20 f=new swing20();
        f.setSize(600,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    JRadioButton r1,r2,r3,r4,r5,r6;
    JLabel l1;
    public swing20()
    {
        r1=new JRadioButton("oracle");
        r2=new JRadioButton("d2k");
        r3=new JRadioButton("dba");
        r4=new JRadioButton("java");
        r5=new JRadioButton("j2ee");
        r6=new JRadioButton("j2me");

        Box v1=Box.createVerticalBox();
        v1.add(r1);v1.add(r2);v1.add(r3);
        //Container con = getContentPane();
        //con.add(v1);
        Box v2=Box.createVerticalBox();
        v2.add(r4);v2.add(r5);v2.add(r6);
        //con.add(v2);

        Box hb=Box.createHorizontalBox();
        hb.add(Box.createHorizontalGlue());
    }
}
```

```
hb.add(v1);
hb.add(Box.createHorizontalGlue());
hb.add(v2);
hb.add(Box.createHorizontalGlue());

l1=new JLabel("selected courses..",JLabel.CENTER);
l1.setBackground(Color.pink);
l1.setForeground(Color.red);
l1.setOpaque(true);

Container con=getContentPane();
con.add("North",hb);
con.add(l1);

r1.addItemListener(this);
r2.addItemListener(this);
r3.addItemListener(this);
r4.addItemListener(this);
r5.addItemListener(this);
r6.addItemListener(this);
}

public void itemStateChanged(ItemEvent i)
{
    String str="";
    if( r1.isSelected())
        str+=r1.getLabel()+" ";
    if( r2.isSelected())
        str+=r2.getLabel()+" ";
    if( r3.isSelected())
        str+=r3.getLabel()+" ";
    if( r4.isSelected())
        str+=r4.getLabel()+" ";
    if( r5.isSelected())
        str+=r5.getLabel()+" ";
    if( r6.isSelected())
        str+=r6.getLabel()+" ";
    l1.setText(str);
}
}
```

```
import javax.swing.*;
import java.awt.*;
public class swing21 extends JApplet
{
    public void init()
    {
        JTextArea ta1=new JTextArea();
        JTextArea ta2=new JTextArea(6,40);
        JTextArea ta3=new JTextArea("welcome to satya");
        JTextArea ta4=new JTextArea("java by rami reddy",6,40);
        JScrollPane jsp=new JScrollPane(ta4);

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(ta1);con.add(ta2);
        con.add(ta3);con.add(jsp);
    }
}
/*
<applet code=swing21 width=800 height=600>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing22 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing22 f=new swing22();
        f.setSize(700,500);
        f.setVisible(true);

        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    JLabel l1,l2,l3;
    JButton b1,b2,b3,b4,b5,b6,b7;
    JTextField tf1,tf2,tf3;
    JTextArea ta;
    public swing22()
    {
        l1=new JLabel("String :");
        l2=new JLabel("St. index :");
        l3=new JLabel("End index :");
        tf1=new JTextField(15);
        tf2=new JTextField(4);
        tf3=new JTextField(4);
        JPanel p1=new JPanel();
        p1.add(l1);p1.add(tf1);
        p1.add(l2);p1.add(tf2);
        p1.add(l3);p1.add(tf3);
    }
}
```

```

ta=new JTextArea();
ta.setForeground(Color.blue);
tf1.setForeground(Color.red);
tf2.setForeground(Color.red);
tf3.setForeground(Color.red);
Font f1=new Font("Arial",Font.BOLD,20);
tf1.setFont(f1);      tf2.setFont(f1);
tf3.setFont(f1);      ta.setFont(f1);
JScrollPane jsp=new
JScrollPane(ta,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

b1=new JButton("insert");
b2=new JButton("replace");
b3=new JButton("append");
b4=new JButton("cut");
b5=new JButton("copy");
b6=new JButton("paste");
b7=new JButton("get info");
 JPanel p2=new JPanel();
 p2.add(b1);p2.add(b2);p2.add(b3);
p2.add(b4);p2.add(b5);p2.add(b6);p2.add(b7);

Container con=getContentPane();
con.add("North",p1);
con.add(jsp);
con.add("South",p2);

b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
b5.addActionListener(this);
b6.addActionListener(this);
b7.addActionListener(this);
}

public void actionPerformed(ActionEvent a)
{
String str=a.getActionCommand();
if(str.equals("insert"))
{
    ta.insert(tf1.getText(),ta.getCaretPosition());
}
else if(str.equals("replace"))
{
    ta.replaceSelection(tf1.getText());
}
}

```

```
        ta.replaceRange(tf1.getText(),Integer.parseInt(tf2.getText()),Integer.parseInt(tf3.getText()));
    }
    else if(str.equals("append"))
    {
        ta.append(tf1.getText());
    }
    else if(str.equals("cut"))
    {
        ta.cut();
    }
    else if(str.equals("copy"))
    {
        ta.copy();
    }
    else if(str.equals("paste"))
    {
        ta.paste();
    }
    else if(str.equals("get info"))
    {
        System.out.println(ta.getText());
        System.out.println(ta.getSelectedText());
        System.out.println(ta.getSelectionStart());
        System.out.println(ta.getSelectionEnd());
        System.out.println(ta.getCaretPosition());
        System.out.println(ta.getTabSize());
        System.out.println(ta.getLineWrap());
        ta.setLineWrap(true);
        ta.setTabSize(20);
        ta.setSelectionColor(Color.yellow);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing23 extends JFrame
implements ComponentListener,MouseListener
{
    public static void main(String args[])
    {
        swing23 f=new swing23();
        f.setBounds(200,100,300,300);
        f.setVisible(true);

        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing23()
    {
        addComponentListener(this);
        addMouseListener(this);
    }
    public void componentShown(ComponentEvent c)
    {
        System.out.println("component shown...");
    }
    public void componentHidden(ComponentEvent c)
    {
        System.out.println("component hidden...");
    }
    public void componentMoved(ComponentEvent c)
    {
        System.out.println("component moved...");
    }
}
```

```
public void componentResized(ComponentEvent c)
{
    System.out.println("component resized...");
}
public void mouseClicked(MouseEvent m)
{
    setVisible(false);
    try{
        Thread.sleep(5000);
    }catch(InterruptedException e){ System.out.println(e); }
    setVisible(true);
}
public void mouseReleased(MouseEvent m){ }
public void mousePressed(MouseEvent m){ }
public void mouseExited(MouseEvent m){ }
public void mouseEntered(MouseEvent m){ }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
public class swing24 extends JPanel implements ActionListener
{
    public static void main(String args[])
    {
        JFrame f=new JFrame();
        Container con=f.getContentPane();
        con.add(new swing24());
        f.setSize(700,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing24()
    {
        JButton b1=new JButton("show dialog..");
        setBackground(Color.cyan);
        add(b1);
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        String str="";
        int r = JOptionPane.showConfirmDialog(this,"do you wish to display time
also?","confirmation",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
```

```
        if(r == 0)
    {
        DateFormat
df=DateFormat.getDateInstance(DateFormat.FULL,DateFormat.FULL);
        str=df.format(new Date());
    }
else if(r==1)
{
    DateFormat df=DateFormat.getDateInstance(DateFormat.FULL);
    str=df.format(new Date());
}
JOptionPane.showMessageDialog(this,str,"date &
time",JOptionPane.INFORMATION_MESSAGE);

    }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
public class swing25 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing25 f=new swing25();
        f.setSize(700,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    public swing25()
    {
        JButton b1=new JButton("show dialog.. ");
        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        String msg=(String) JOptionPane.showInputDialog(this,"enter your name :");
        JOptionPane.showMessageDialog(this,"hello "+msg);
    }
}
```

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing26 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing26 f=new swing26();
        f.setSize(700,400);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    Container con;
    public swing26()
    {
        JButton b1=new JButton("show dialog..");

        con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        String colors[]={ "red","blue","green","yellow","magenta","pink"};
        String msg=(String) JOptionPane.showInputDialog(this,"enter your name
        :","colors",JOptionPane.WARNING_MESSAGE,null,colors,colors[2]);
        if(msg !=null)
        {
            if(msg.equals("red"))
                con.setBackground(Color.red);
            else if(msg.equals("blue"))

```

```
        con.setBackground(Color.blue);
    else if(msg.equals("green"))
        con.setBackground(Color.green);
    else if(msg.equals("magenta"))
        con.setBackground(Color.magenta);
    else if(msg.equals("yellow"))
        con.setBackground(Color.yellow);
    else if(msg.equals("pink"))
        con.setBackground(Color.pink);
    }
}

}
```

ramireddy.satyam

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing27 extends JFrame
implements ActionListener,ContainerListener
{
    public static void main(String args[])
    {
        swing27 f=new swing27();
        f.setSize(700,500);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    JButton b1,b2,b3;
    Container con;
    public swing27()
    {
        b1=new JButton("add");
        b2=new JButton("remove");
        b3=new JButton("welcome",new ImageIcon("alice.gif"));

        con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);con.add(b2);con.add(b3);

        b1.addActionListener(this);
        b2.addActionListener(this);
        con.addContainerListener(this);
    }
    public void actionPerformed(ActionEvent a)
```

```
{  
    String str=a.getActionCommand();  
    if(str.equals("add"))  
        con.add(b3);  
    else  
        con.remove(b3);  
    repaint();  
}  
public void componentAdded(ContainerEvent c)  
{  
    con.setBackground(Color.pink);  
}  
public void componentRemoved(ContainerEvent c)  
{  
    con.setBackground(Color.orange);  
}  
}
```

```
import javax.swing.*;
import java.awt.event.*;
public class swing28 extends JApplet
implements KeyListener
{
    public void init()
    {
        addKeyListener(this);
    }
    public void keyPressed(KeyEvent k)
    {
        System.out.println("key pressed :" +k.getKeyCode());
    }
    public void keyReleased(KeyEvent k)
    {
        System.out.println("key released..");
    }
    public void keyTyped(KeyEvent k)
    {
        System.out.println("key typed :" +k.getKeyChar());
    }
}
/*
<applet code=swing28 width=200 height=200>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swing29 extends JFrame
{
    public static void main(String args[])
    {
        swing29 f=new swing29();
        f.setSize(400,300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    Container con;
    public swing29()
    {
        con=getContentPane();
        addKeyListener(new mykey());
    }
    public class mykey extends KeyAdapter
    {
        public void keyPressed(KeyEvent k)
        {
            switch(k.getKeyCode())
            {
                case KeyEvent.VK_F1 : con.setBackground(Color.red);break;
                case KeyEvent.VK_F2 : con.setBackground(Color.magenta);break;
                case KeyEvent.VK_F3 : con.setBackground(Color.orange);break;
                case KeyEvent.VK_F4 : con.setBackground(Color.pink);break;
                case KeyEvent.VK_ESCAPE : System.exit(0);
            }
        }
    }
}
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swing30 extends JFrame
implements WindowListener
{
    public static void main(String args[])
    {
        swing30 f=new swing30();
        f.setBounds(300,200,300,300);
        f.setVisible(true);
    }
    public swing30()
    {
        addWindowListener(this);
    }
    public void windowClosing(WindowEvent w)
    {
        System.out.println("window closing ...");
        System.exit(0);
    }
    public void windowClosed(WindowEvent w)
    {
        System.out.println("window closed...");
    }
    public void windowOpened(WindowEvent w)
    {
        System.out.println("window opened ...");
    }
    public void windowDeiconified(WindowEvent w)
    {
        System.out.println("window deiconified ...");
    }
}
```

```
public void windowIconified(WindowEvent w)
{
    System.out.println("window iconified...");
}
public void windowDeactivated(WindowEvent w)
{
    System.out.println("window deactivated...");
}
public void windowActivated(WindowEvent w)
{
    System.out.println("window activated...");
}
}
```

ramireddy.satya

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.event.*;
public class swing31 extends JPanel
implements ChangeListener
{ public static void main(String args[])
{
    JFrame f1=new JFrame();
    //f1.getContentPane().add(new swing31());
    Container con=f1.getContentPane();
    con.add(new swing31());
    f1.setSize(600,400);
    f1.setVisible(true);
    f1.addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent w)
        {
            System.exit(0);
        }
    });
}
JSlider s1,s2;
JTextField tf;
public swing31()
{
    s1=new JSlider(JSeparator.HORIZONTAL);
    s2=new JSlider(JSeparator.HORIZONTAL,0,200,25);
    s2.setPaintTicks(true);
    s2.setPaintLabels(true);
    s2.setMajorTickSpacing(50);
    s2.setMinorTickSpacing(5);
    s2.setBorder(new BevelBorder(BevelBorder.RAISED,Color.pink,Color.red));

    Box vb=Box.createVerticalBox();
    vb.add(Box.createVerticalGlue());
    vb.add(s1);
```

```
vb.add(Box.createVerticalGlue());
vb.add(s2);
vb.add(Box.createVerticalGlue());

tf=new JTextField();
tf.setForeground(Color.red);
tf.setFont(new Font("Arial",Font.BOLD,20));

setLayout(new BorderLayout());
add(vb);
add("South",tf);

s2.addChangeListener(this);
}

public void stateChanged(ChangeEvent c)
{
    tf.setText(String.valueOf(s2.getValue()));
}
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class swing32 extends JPanel
implements ActionListener
{ public static void main(String args[])
{
    JFrame f1=new JFrame();
    f1.getContentPane().add(new swing32());
    f1.setSize(600,400);
    f1.setVisible(true);
    f1.addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent w)
        {
            System.exit(0);
        }
    });
    JButton b1,b2,b3;
    JLabel l1;
    javax.swing.Timer t1;
public swing32()
{
    l1=new JLabel(new Date().toString(),JLabel.CENTER);
    b1=new JButton("start");
    b2=new JButton("stop");
    b3=new JButton("restart");
    JPanel p1=new JPanel();
    p1.add(b1);p1.add(b2);p1.add(b3);

    setLayout(new BorderLayout());
    add(l1);
    add("South",p1);

    t1=new javax.swing.Timer(1000,new myaction());
    t1.setInitialDelay(5000);
}
```

```
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
}
public void actionPerformed(ActionEvent a)
{
    JButton b=(JButton)a.getSource();
    if(b==b1)
        t1.start();
    else if(b==b2)
        t1.stop();
    else if(b==b3)
        t1.restart();
}
int i=0;
public class myaction implements ActionListener
{
    Color colors[]=new
Color[]{Color.red,Color.blue,Color.orange,Color.green,Color.magenta,Color.pink};
    public void actionPerformed(ActionEvent a)
    {
        tl.setText(new Date().toString());
        tl.setForeground(colors[i]);
        i++;
        if(i == colors.length)
            i=0;
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class swing33 extends JPanel
implements ActionListener
{ public static void main(String args[])
{
    JFrame f1=new JFrame();
    f1.getContentPane().add(new swing33());
    f1.setSize(600,400);
    f1.setVisible(true);
    f1.addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent w)
        {
            System.exit(0);
        }
    });
}
JButton b1;
JTextField tf1;
JColorChooser jc;
JDialog jd;
public swing33()
{
    b1=new JButton("show color dialog..");
    tf1=new JTextField(" java course by rami reddy ");
    tf1.setFont(new Font("Arial",Font.BOLD,30));

    setLayout(new BorderLayout());
    add(tf1);
    add("South",b1);

    jc=new JColorChooser();
    jd=jc.createDialog(this,"color dialog",
        true,jc,this,this);
```

```
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {
        String str=a.getActionCommand();
        if(str.equals("OK"))
        {
            tf1.setForeground(jc.getColor());
            jd.dispose();
        }
        else if(str.equals("Cancel"))
        {
            jd.dispose();
        }
        else if(str.equals("show color dialog.."))
        {
            jd.show();
        }
    }
}
```

```
import javax.swing.*;
import java.awt.*;
public class swing34 extends JApplet
{
    public void init()
    {   swing24 p1=new swing24();
        swing31 p2=new swing31();
        swing32 p3=new swing32();
        swing33 p4=new swing33();
        //JTabbedPane jt=new JTabbedPane();
        JTabbedPane jt=new JTabbedPane(JTabbedPane.LEFT);
        jt.addTab("dialogs",new ImageIcon("paste.gif"),p1,"dialogs demo");
        jt.addTab("sliders",null,p2,"sliders demo");
        jt.addTab("timer",null,p3,"timer demo");
        jt.insertTab("colors",null,p4,"color chooser demo",1);

        Container con=getContentPane();
        con.add(jt);
    }
}
/*
<applet code=swing34 width=600 height=400>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.*;
public class swing35 extends JApplet
{
    public void init()
    {   swing24 p1=new swing24();
        swing31 p2=new swing31();
        swing32 p3=new swing32();
        JSplitPane jsp=new JSplitPane(JSplitPane.VERTICAL_SPLIT,p1,p2);
        JSplitPane jsp1=new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
        jsp1.setLeftComponent(p3);
        jsp1.setRightComponent(jsp);
        Container con=getContentPane();
        con.add(jsp1);
    }
}
/*
<applet code=swing35 width=600 height=400>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.io.*;
public class swing36 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing36 f=new swing36();
        f.setSize(800,550);
        f.setVisible(true);

        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    JMenuItem m11,m12,m13,m14,m21,m22,m23;
    JCheckBoxMenuItem m31,m32;
    JRadioButtonMenuItem f1,f2,f3,b1,b2,b3;
    JTextArea ta;
    JFileChooser fc;
    public swing36()
    {
        JMenu m1=new JMenu("File");
        m1.setIcon(new ImageIcon("help.gif"));
        m1.setMnemonic('f');
        m11=new JMenuItem("New");
        m12=new JMenuItem("Open",new ImageIcon("open.gif"));
        m13=new JMenuItem("Save",new ImageIcon("save.gif"));
        m14=new JMenuItem("Exit",'e');
        m1.add(m11);
        m1.addSeparator();
```

```

m1.add(m12);m1.add(m13);
m1.addSeparator();
m1.add(m14);

JMenu m2=new JMenu("Edit");
m21=new JMenuItem("cut",new ImageIcon("cut.gif"));
m22=new JMenuItem("copy",new ImageIcon("copy.gif"));
m23=new JMenuItem("paste",new ImageIcon("paste.gif"));
m2.add(m21);m2.add(m22);m2.add(m23);

JMenu m3=new JMenu("Styles");
m31=new JCheckBoxMenuItem("Bold");
m32=new JCheckBoxMenuItem("Italic");
m3.add(m31);m3.add(m32);

JMenu m4=new JMenu("Colors");

JMenu m41=new JMenu("Fore Colors");
f1=new JRadioButtonMenuItem("red");
f2=new JRadioButtonMenuItem("magenta");
f3=new JRadioButtonMenuItem("blue");
m41.add(f1);m41.add(f2);m41.add(f3);

JMenu m42=new JMenu("Back Colors");
b1=new JRadioButtonMenuItem("cyan");
b2=new JRadioButtonMenuItem("pink");
b3=new JRadioButtonMenuItem("orange");
m42.add(b1); m42.add(b2); m42.add(b3);
m4.add(m41);m4.add(m42);

JMenuBar mb=new JMenuBar();
mb.add(m1);mb.add(m2);mb.add(m3);
mb.add(m4);

ta=new JTextArea();
ta.setFont(new Font("Arial",Font.BOLD,20));
ta.setForeground(Color.blue);
JScrollPane jsp=new
JScrollPane(ta,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
Container con=getContentPane();
con.add("North",mb);
con.add(jsp);

ButtonGroup bg1=new ButtonGroup();
bg1.add(f1);bg1.add(f2);bg1.add(f3);

```

```

ButtonGroup bg2=new ButtonGroup();
bg2.add(b1);bg2.add(b2);bg2.add(b3);

m11.addActionListener(this);
m12.addActionListener(this);
m13.addActionListener(this);
m14.addActionListener(this);

fc=new JFileChooser();
}

public void actionPerformed(ActionEvent a)
{ try{
    JMenuItem m=(JMenuItem)a.getSource();
    if(m==m11)
        ta.setText("");
    else if(m==m14)
        System.exit(0);
    else if(m==m12)
    {
        int r=fc.showOpenDialog(this);
        if(r==JFileChooser.APPROVE_OPTION)
        {
            File f1=fc.getSelectedFile();
            BufferedReader br=new BufferedReader(new FileReader(f1));
            String str="";
            while((str=br.readLine())!= null)
            {
                ta.append(str+"\n");
            }
            br.close();
        }
        else if(r==JFileChooser.CANCEL_OPTION)
        {
            JOptionPane.showMessageDialog(this,"cancel button is clicked..");
        }
    }
    else if(m==m13)
    {
        int r=fc.showSaveDialog(this);
        if(r==JFileChooser.APPROVE_OPTION)
        {
            File f1=fc.getSelectedFile();
            FileWriter fp=new FileWriter(f1);
            fp.write(ta.getText());
            fp.close();
            JOptionPane.showMessageDialog(this,"file saved");
        }
    }
}
}

```

```
        }
    else if(r==JFileChooser.CANCEL_OPTION)
    {
        JOptionPane.showMessageDialog(this,"cancel button is clicked..");
    }
}catch(IOException e)
{
    System.out.println(e);
}
}
```

ramireddy.Satya

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swing37 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing37 f=new swing37();
        f.setSize(700,450);
        f.setVisible(true);
    }
    JButton b1,b2,b3;
    JTextArea ta;
    JMenuItem m1,m2,m3;
    JPopupMenu pm;
    public swing37()
    {
        addWindowListener(new mywindow());
        ta=new JTextArea();
        ta.setFont(new Font("Arial",Font.BOLD,20));
        ta.setForeground(Color.blue);
        JScrollPane jsp=new
JScrollPane(ta,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        JToolBar jt=new JToolBar();
        jt.setFloatable(false);
        jt.add(b1=new JButton(new ImageIcon("cut.gif")));
        jt.add(b2=new JButton(new ImageIcon("copy.gif")));
        jt.addSeparator();
        jt.add(b3=new JButton(new ImageIcon("paste.gif")));
        Container con=getContentPane();
        con.add("South",jt);
        con.add(jsp);
        pm=new JPopupMenu();
        m1=new JMenuItem("cut");
        m2=new JMenuItem("copy");
        m3=new JMenuItem("paste");
```

```

        pm.add(m1);pm.add(m2);pm.add(m3);
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        m1.addActionListener(this);
        m2.addActionListener(this);
        m3.addActionListener(this);
        ta.addMouseListener(new mymouse());
    }
    public void actionPerformed(ActionEvent a)
    {
        Object obj=a.getSource();
        if( obj instanceof JButton)
        {
            JButton b=(JButton)obj;
            if(b==b1)
                ta.cut();
            else if(b==b2)
                ta.copy();
            else if(b==b3)
                ta.paste();
        }else if( obj instanceof JMenuItem)
        {
            JMenuItem m=(JMenuItem)obj;
            if(m==m1)
                ta.cut();
            else if(m==m2)
                ta.copy();
            else if(m==m3)
                ta.paste();
        }
    }
    public class mymouse extends MouseAdapter
    {
        public void mouseReleased(MouseEvent m)
        {
            if(m.isPopupTrigger())
            {
                pm.show(m.getComponent(),m.getX(),m.getY());
            }
        }
    }
    public class mywindow extends WindowAdapter
    {
        public void windowClosing(WindowEvent w)
        {

```

```
        System.exit(0);
    }
}
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing38 extends JApplet
{
    Canvas c1;
    JButton b1;
    public void init()
    {
        b1=new JButton("demo button");
        c1=new Canvas();
        c1.setBackground(Color.white);
        c1.setSize(300,300);

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(b1);con.add(c1);

        mymouse m1=new mymouse();
        c1.addMouseListener(m1);
        c1.addMouseMotionListener(m1);
    }
    int x1,y1,x2,y2;
    public class mymouse extends MouseAdapter implements MouseMotionListener
    {
        public void mousePressed(MouseEvent m)
        {
            x1=m.getX();
            y1=m.getY();
        }
        public void mouseDragged(MouseEvent m)
        {
            x2=m.getX();
            y2=m.getY();
            Graphics g=c1.getGraphics();
            g.setColor(Color.red);
            g.drawLine(x1,y1,x2,y2);
        }
    }
}
```

```
    x1=x2;
    y1=y2;
}
public void mouseMoved(MouseEvent m){}

}
/*
<applet code=swing38 width=600 height=400>
</applet>
*/
```

ramireddy.satyam

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing39 extends JApplet
implements AdjustmentListener
{   JScrollBar s1,s2;
    JTextField tf;
    public void init()
    {
        s1=new JScrollBar();
        s2=new JScrollBar(JScrollBar.HORIZONTAL,200,0,0,1000);
        s2.setUnitIncrement(10);
        s2.setBlockIncrement(50);
        s1.setBounds(20,50,30,200);
        s2.setBounds(50,20,200,30);
        tf=new JTextField();
        tf.setBounds(100,100,100,30);
        Container con=getContentPane();
        con.setLayout(null);
        con.add(s1);con.add(s2);con.add(tf);

        s1.addAdjustmentListener(this);
        s2.addAdjustmentListener(this);
    }
    public void adjustmentValueChanged(AdjustmentEvent a)
    {
        JScrollBar s=(JScrollBar)a.getSource();
        tf.setText(String.valueOf(s.getValue()));
    }
}
/*
<applet code=swing39 width=500 height=400>
</applet>
*/
```

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing40 extends JApplet
{
    JTable jt;
    public void init()
    {
        String columns[]={"student number","student name","student courses"};
        String data[][]={{ {"1001","ram","java,oracle"}, {"1002","sam","c,c++,java"}, {"1003","sita","j2ee,j2me"}, {"1004","raju","java,dba,d2k"} }};
        jt=new JTable(data,columns);
        jt.setSelectionForeground(Color.red);
        jt.setSelectionBackground(Color.yellow);
        JScrollPane jsp=new JScrollPane(jt);
        Container con=getContentPane();
        con.add(jsp);
        jt.addMouseListener(new mymouse());
    }
    public class mymouse extends MouseAdapter
    {
        public void mouseClicked(MouseEvent m)
        {
            int rows=jt.getRowCount();
            int cols=jt.getColumnCount();
            int r=jt.getSelectedRow();
            int c=jt.getSelectedColumn();
            System.out.println("rows :" +rows);
            System.out.println("columns :" +cols);
            System.out.println("selected row :" +r);
            System.out.println("selected column :" +c);
            System.out.println("COLUMN NAMES :");
            for(int i=0;i<cols;i++)
            System.out.println(jt.getColumnName(i));
            showStatus("selected value :" +jt.getValueAt(r,c));
        }
    }
}

```

```

/*
<applet code=swing40 width=500 height=400>
</applet>
*/



import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import javax.swing.tree.*;
public class swing41 extends JApplet
implements TreeSelectionListener
{
    JTree jt;
    JTextField tf;
    public void init()
    {
        DefaultMutableTreeNode n=new DefaultMutableTreeNode("java topics");
        DefaultMutableTreeNode n1=new DefaultMutableTreeNode("coretopics");
        DefaultMutableTreeNode n2=new DefaultMutableTreeNode("adv.topics");
        DefaultMutableTreeNode n11=new DefaultMutableTreeNode("interfaces");
        DefaultMutableTreeNode n12=new DefaultMutableTreeNode("packages");
        DefaultMutableTreeNode n13=new DefaultMutableTreeNode("exceptions");
        DefaultMutableTreeNode n14=new DefaultMutableTreeNode("iostreams");
        DefaultMutableTreeNode n15=new DefaultMutableTreeNode("threads");
        DefaultMutableTreeNode n21=new DefaultMutableTreeNode("swings");
        DefaultMutableTreeNode n22=new DefaultMutableTreeNode("jdbc");
        DefaultMutableTreeNode n23=new DefaultMutableTreeNode("servlets");
        DefaultMutableTreeNode n24=new DefaultMutableTreeNode("jsp");
        DefaultMutableTreeNode n25=new DefaultMutableTreeNode("rmi");
        DefaultMutableTreeNode n26=new DefaultMutableTreeNode("sockets");

        n1.add(n11);n1.add(n12);n1.add(n13);n1.add(n14);n1.add(n15);
        n2.add(n21);n2.add(n22);n2.add(n23);n2.add(n24);n2.add(n25);n2.add(n26);
        n.add(n1);n.add(n2);

        jt=new JTree(n);
        JScrollPane jsp=new JScrollPane(jt);
        //jt.setRowHeight(50);
        //jt.setEditable(true);
        DefaultTreeCellRenderer r=(DefaultTreeCellRenderer)jt.getCellRenderer();
        r.setLeafIcon(new ImageIcon("bullet.gif"));
        r.setOpenIcon(new ImageIcon("open.gif"));
        r.setClosedIcon(new ImageIcon("paste.gif"));
        r.setTextSelectionColor(Color.red);
    }
}

```

```
r.setTextNonSelectionColor(Color.blue);
r.setBackgroundSelectionColor(Color.yellow);

tf=new JTextField();

Container con=getContentPane();
con.add(jsp);
con.add("South",tf);

jt.addTreeSelectionListener(this);
}

public void valueChanged(TreeSelectionEvent e)
{
    TreePath tp=e.getPath();
    tf.setText(tp.toString());
    Object obj[] =tp.getPath();
    for(int i=0;i<obj.length;i++)
        System.out.println(obj[i]);
}

/*
<applet code=swing41 width=500 height=400>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing42 extends JApplet
implements ActionListener
{    JComboBox jc;
    JButton b1;
    public void init()
    {        String weeks[]={ "Sunday","Monday","Wednesday","Thursday","Friday" };
        jc=new JComboBox(weeks);
        jc.addItem("Saturday");
        jc.insertItemAt("Tuesday",2);
        jc.setMaximumRowCount(4);

        b1=new JButton("remove");

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(jc);con.add(b1);

        jc.addActionListener(this);
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent a)
    {
        Object obj=a.getSource();
        if(obj instanceof JComboBox)
        {
            System.out.println(jc.getItemCount());
            System.out.println(jc.getSelectedIndex());
            System.out.println(jc.getSelectedItem());
            System.out.println(jc.getItemAt(4));
            System.out.println(jc.getMaximumRowCount());
        }
    }
}
```

```
        }
        else if(obj instanceof JButton)
        {
            jc.removeAllItems();
            // jc.removeItem("Tuesday");
            // jc.removeItemAt(2);
        }
    }
/*
<applet code=swing42 width=500 height=400>
</applet>
*/
```

ramireddy.satya

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing43 extends JApplet
{
    JComboBox jc;
    Icon icons[]={new ImageIcon("cut.gif"),new ImageIcon("copy.gif"),new
    ImageIcon("paste.gif"),new ImageIcon("new.gif"),new ImageIcon("open.gif"),new
    ImageIcon("save.gif"),new ImageIcon("help.gif")};
    public void init()
    {
        String
weeks[]={"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"
    };
        jc=new JComboBox(weeks);
        jc.setMaximumRowCount(4);
        jc.setPreferredSize(new Dimension(200,40));

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(jc);

        jc.setRenderer(new democlass());
    }
    public class democlass extends JLabel implements ListCellRenderer
    {
        public democlass()
        {
            setOpaque(true);
        }
        public Component getListCellRendererComponent(JList j1,Object value,int
index,boolean isSelected,boolean hasfocus)
        {
            if( value !=null)
                setText((String)value);
            if( index >= 0 && index <= icons.length)
                setIcon(icons[index]);
            if(isSelected)
            {
                setBackground(Color.yellow);
            }
        }
    }
}

```

```
        setForeground(Color.red);
    }
else
{
    setBackground(Color.cyan);
    setForeground(Color.blue);
}
return this;
}
}
/*
<applet code=swing43 width=500 height=400>
</applet>
*/
```

ramireddy.Satya

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing44 extends JApplet
{   JList list1;
    Icon icons[]={new ImageIcon("cut.gif"),new ImageIcon("copy.gif"),new
ImageIcon("paste.gif"),new ImageIcon("new.gif"),new ImageIcon("open.gif"),new
ImageIcon("save.gif"),new ImageIcon("help.gif")};
    public void init()
    {
        String
weeks[]={"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"
};
        list1=new JList(weeks);
        JScrollPane jsp=new JScrollPane(list1);
        jsp.setPreferredSize(new Dimension(200,150));

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(jsp);

        list1.setCellRenderer(new democlass());
    }
    public class democlass extends JLabel implements ListCellRenderer
    {
        public democlass()
        {
            setOpaque(true);
        }
        public Component getListCellRendererComponent(JList j1, Object value,int
index,boolean isSelected,boolean hasfocus)
        {
            if( value !=null)
                setText((String)value);
            if( index >= 0 && index <= icons.length)
                setIcon(icons[index]);
            if(isSelected)
            {
```

```
        setBackground(Color.yellow);
        setForeground(Color.red);
    }
else
{
    setBackground(Color.cyan);
    setForeground(Color.blue);
}
return this;
}
}
/*
<applet code=swing44 width=500 height=400>
</applet>
*/
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing45 extends JApplet
{
    JList list1; JCheckBox c1;
    JButton b1;
    public void init()
    {
        String
courses[]={ "oracle","java","j2ee","j2me","dba","c++","vc++","d2k","vb.net","asp.net"};
        list1=new JList(courses);
        JScrollPane jsp=new JScrollPane(list1);
        jsp.setPreferredSize(new Dimension(200,150));
        list1.setFont(new Font("Arial",Font.BOLD,20));
        list1.setForeground(Color.blue);

        c1=new JCheckBox("multiple",true);
        b1=new JButton("get item(s)");

        Container con=getContentPane();
        con.setLayout(new FlowLayout());
        con.add(jsp);con.add(c1);con.add(b1);

        b1.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent a)
            {
                if( c1.isSelected())
                {
                    Object items[]=list1.getSelectedValues();
                    int n[]=list1.getSelectedIndices();
                    System.out.println("MULTIPLE SELECTIONS..");
                    for(int i=0;i<items.length;i++)
                        System.out.println(n[i]+": "+items[i]);
                }
                else
                {

```

```
        System.out.println("SINGLE SELECTION...");  
        System.out.println(list1.getSelectedIndex()+" :  
"+list1.getSelectedItem());  
    }  
});  
c1.addItemListener(new ItemListener()  
{  
    public void itemStateChanged(ItemEvent i)  
    {  
        if(c1.isSelected())  
        {  
  
list1.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);  
        }  
        else  
        {  
  
list1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);  
        }  
        list1.clearSelection();  
    }  
});  
}  
/*  
<applet code=swing45 width=600 height=400>  
</applet>  
*/
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing46 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing46 f=new swing46();
        f.setSize(700,450);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    JButton b1,b2,b3,b4;
    JComboBox c;
    JPanel mp;
    CardLayout cl;
    public swing46()
    {
        b1=new JButton("first");
        b2=new JButton("previous");
        b3=new JButton("next");
        b4=new JButton("last");
        String items[]={ "card1","card2","card3","card4"};
        c=new JComboBox(items);
        JPanel tp=new JPanel();
        tp.add(b1);tp.add(b2);tp.add(b3);tp.add(b4);tp.add(c);

        mp=new JPanel();
        cl=new CardLayout();
        mp.setLayout(cl);
    }
}
```

```
swing24 c1=new swing24();
swing31 c2=new swing31();
swing32 c3=new swing32();
swing33 c4=new swing33();
mp.add(c1,"card1");
mp.add(c2,"card2");
mp.add(c3,"card3");
mp.add(c4,"card4");

Container con=getContentPane();
con.add("North",tp);
con.add(mp);

b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
c.addActionListener(this);
}

public void actionPerformed(ActionEvent a)
{
    Object obj=a.getSource();
    if(obj instanceof JButton)
    {
        JButton b=(JButton)obj;
        if(b==b1)
            cl.first(mp);
        else if(b==b2)
            cl.previous(mp);
        else if(b==b3)
            cl.next(mp);
        else if(b==b4)
            cl.last(mp);
    }
    else
    {
        cl.show(mp, (String)c.getSelectedItem() );
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing47 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing47 f=new swing47();
        f.setSize(220,150);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }
    JProgressBar p1;
    JButton b1,b2;
    public swing47()
    {
        p1=new JProgressBar();
        b1=new JButton("start");
        b2=new JButton("stop");
        p1.setBounds(10,10,200,40);
        b1.setBounds(10,70,90,30);
        b2.setBounds(110,70,90,30);
        Container con=getContentPane();
        con.setLayout(null);
        con.add(p1);con.add(b1);con.add(b2);

        b1.addActionListener(this);
        b2.addActionListener(this);
    }
    demothread t1;
    public void actionPerformed(ActionEvent a)
```

```
{  
    JButton b=(JButton)a.getSource();  
    if(b==b1)  
    {  
        if( t1 ==null || ! t1.isAlive())  
        {  
            t1=new demothread();  
            t1.start();  
        }  
    }  
    else if(b==b2)  
    {  
        t1.flag=true;  
    }  
}
```

```
class demothread extends Thread  
{  
    boolean flag;  
    public demothread()  
    {  
        super("child");  
    }  
    public void run()  
    {  
        p1.setMaximum(200);  
        p1.setMinimum(0);  
        for(int i=0;i<=200;i+=2)  
        {  
            p1.setValue(i);  
            try{  
                Thread.sleep(300);  
            }catch(Exception e)  
            {  
                System.out.println(e);  
            }  
            if( flag)  
                break;  
        }  
    }  
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing48 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing48 f=new swing48();
        f.setSize(220,150);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    JButton b1,b2;
    Container con;
    public swing48()
    {
        b1=new JButton("start");
        b2=new JButton("stop");
        b1.setBounds(10,70,90,30);
        b2.setBounds(110,70,90,30);
        con=getContentPane();
        con.setLayout(null);
        con.add(b1);con.add(b2);

        b1.addActionListener(this);
        b2.addActionListener(this);
    }
    demothread t1;
    public void actionPerformed(ActionEvent a)
    {
        JButton b=(JButton)a.getSource();
```

```

if(b==b1)
{
    if( t1 ==null || ! t1.isAlive())
    {
        t1=new demothread();
        t1.start();
    }
}
else if(b==b2)
{
    t1.flag=true;
}
}

```

```

class demothread extends Thread
{
    boolean flag;
    ProgressMonitor pm;
    public demothread()
    {
        super("child");
        pm=new ProgressMonitor(con,"our progress","progress",0,200);
    }
    public void run()
    {
        for(int i=0;i<=200;i+=2)
        {
            pm.setNote(i/2+"% in progress..");
            pm.setProgress(i);
            try{
                Thread.sleep(300);
            }catch(Exception e)
            {
                System.out.println(e);
            }
            if( flag)
                break;
        }
        pm.close();
    }
}

```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class swing49 extends JFrame
implements ActionListener
{
    public static void main(String args[])
    {
        swing49 f=new swing49();
        f.setSize(700,550);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

JMenuItem m1,m2,m3,m4;
JInternalFrame f1;
JDesktopPane jd;
public swing49()
{
    JMenu m=new JMenu("Demos");
    m.add(m1=new JMenuItem("dialogs"));
    m.add(m2=new JMenuItem("slider"));
    m.add(m3=new JMenuItem("timer"));
    m.add(m4=new JMenuItem("colos"));

    JMenuBar mb=new JMenuBar();
    mb.add(m);

    jd=new JDesktopPane();

    Container con=getContentPane();
    con.add("North",mb);
```

```
con.add(jd);

m1.addActionListener(this);
m2.addActionListener(this);
m3.addActionListener(this);
m4.addActionListener(this);
}

public void actionPerformed(ActionEvent a)
{
    f1=new JInternalFrame("ourframe",true,true,true,true);
    Container c1=f1.getContentPane();
    JMenuItem m=(JMenuItem)a.getSource();
    if(m==m1)
        c1.add(new swing24());
    else if(m==m2)
        c1.add(new swing31());
    else if(m==m3)
        c1.add(new swing32());
    else if(m==m4)
        c1.add(new swing33());
    jd.add(f1);
    f1.setBounds(20,20,400,300);
    f1.show();
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.plaf.windows.*;
import com.sun.java.swing.plaf.motif.*;
import javax.swing.plaf.metal.*;
public class swing50 extends JFrame
implements ActionListener
{
    WindowsLookAndFeel f1=new WindowsLookAndFeel();
    MotifLookAndFeel f2=new MotifLookAndFeel();
    MetalLookAndFeel f3=new MetalLookAndFeel();

    public static void main(String args[])
    {
        swing50 f=new swing50();
        f.setSize(700,550);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent w)
            {
                System.exit(0);
            }
        });
    }

    JMenuItem m1,m2,m3;

    public swing50()
    {
        JMenu m=new JMenu("Look & Feel");
        m.add(m1=new JMenuItem("Windows"));
        m.add(m2=new JMenuItem("Motif"));
        m.add(m3=new JMenuItem("Metal"));

        JMenuBar mb=new JMenuBar();
        mb.add(m);
    }
}
```

```
JPanel p1=new JPanel();
p1.add(new JButton("demobutton"));
p1.add(new JLabel("demo label"));
p1.add(new JCheckBox("demo box"));
p1.add(new JRadioButton("radio button"));
p1.add(new JToggleButton("toggle button"));
p1.add(new JTextField("welcome",20));
p1.add(new JSlider());
p1.add(new JScrollPane());
String weeks[]={ "Sunday","Monday","Tuesday"};
p1.add(new JComboBox(weeks));

Container con=getContentPane();
con.add("North",mb);
con.add(p1);

m1.addActionListener(this);
m2.addActionListener(this);
m3.addActionListener(this);

}

public void actionPerformed(ActionEvent a)
{ try{
    JMenuItem m=(JMenuItem)a.getSource();
    if(m==m1)
        UIManager.setLookAndFeel(f1);
    else if(m==m2)
        UIManager.setLookAndFeel(f2);
    else if(m==m3)
        UIManager.setLookAndFeel(f3);
    SwingUtilities.updateComponentTreeUI(this);
} catch(UnsupportedLookAndFeelException e)
{
    System.out.println(e);
}
}
```

```
import javax.swing.*;
import java.awt.*;

class swingdemo extends Frame
{
    swingdemo()
    {
        Button b1 = new Button("OK");
        add(b1,"South");
        JButton b2 = new JButton("CANCEL");
        add(b2,"North");
        setSize(300,400);
        setVisible(true);
    }
    public static void main(String[] args)
    {
        new swingdemo();
    }
}
```

```
/*
<applet code=TJMenu width=500 height=600>
</applet>
*/
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

public class TJMenu extends JApplet
{
    public void init()
    {
        Container con = this.getContentPane();
        JMenuBar mb = new JMenuBar();
        mb.setBorder(new BevelBorder(BevelBorder.RAISED));
        mb.setBorderPainted(true);
        con.add(mb, BorderLayout.NORTH);
        JMenu filemenu = new JMenu("File",true);
        mb.add(filemenu);

        filemenu.add(new JMenuItem("New"));
        filemenu.add(new JMenuItem("Open"));
        filemenu.addSeparator();
        filemenu.add(new JMenuItem("Save"));
        filemenu.add(new JMenuItem("Save as"));
        filemenu.addSeparator();

        JMenu editmenu = new JMenu("Edit");
        mb.add(editmenu);
        editmenu.add(new JMenuItem("Undo"));
        editmenu.addSeparator();
        editmenu.add(new JMenuItem("Cut"));
        editmenu.add(new JMenuItem("Copy"));
        editmenu.add(new JMenuItem("Paste"));

        JMenu optionsmenu = new JMenu("Options");
    }
}
```

```
mb.add(optionsmenu);

JMenu bookmarksmenu = new JMenu("Book Marks");
optionsmenu.add(bookmarksmenu);

JMenuItem addMI = new JMenuItem("Add Alt-k");
bookmarksmenu.add(addMI);
JMenuItem editMI = new JMenuItem("Edit Alt-B");
bookmarksmenu.add(editMI);
JMenu guidemenu = new JMenu("guide");
bookmarksmenu.add(guidemenu);

JMenuItem whatisnewMI = new JMenuItem("whats New");
whatisnewMI.setMnemonic('e');
guidemenu.add(whatisnewMI);

JMenuItem whatiscoolMI = new JMenuItem("whats cool");
whatiscoolMI.setMnemonic('C');
guidemenu.add(whatiscoolMI);

JMenuItem javaconsolemi = new JMenuItem("java console");
optionsmenu.add(javaconsolemi);

JMenuItem addressbookmi = new JMenuItem("address book");
optionsmenu.add(addressbookmi);
}
```

```
/*
<applet code=TJMenu width=500 height=600>
</applet>
*/
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event;

public class TJMenu extends JApplet
{
    public void init()
    {
        Container con = this.getContentPane();
        JMenuBar mb = new JMenuBar();
        mb.setBorder(new BevelBorder(BevelBorder.RAISED));
        mb.setBorderPainted(true);
        con.add(mb, BorderLayout.NORTH);
        JMenu filemenu = new JMenu("File",true);
        mb.add(filemenu);

        filemenu.add(new JMenuItem("New"));
        filemenu.add(new JMenuItem("Open"));
        filemenu.addSeparator();
        filemenu.add(new JMenuItem("Save"));
        filemenu.add(new JMenuItem("Save as"));
        filemenu.addSeparator();

        JMenu editmenu = new JMenu("Edit");
        mb.add(editmenu);
        editmenu.add(new JMenuItem("Undo"));
        editmenu.addSeparator();
        editmenu.add(new JMenuItem("Cut"));
        editmenu.add(new JMenuItem("Copy"));
        editmenu.add(new JMenuItem("Paste"));
    }
}
```

```
JMenu optionsmenu = new JMenu("Options");
mb.add(optionsmenu);

JMenu bookmarksmenu = new JMenu("Book Marks");
optionsmenu.add(bookmarksmenu);

JMenuItem addMI = new JMenuItem("Add Alt-k");
bookmarksmenu.add(addMI);
JMenuItem editMI = new JMenuItem("Edit Alt-B");
bookmarksmenu.add(editMI);
JMenu guidemenu = new JMenu("guide");
bookmarksmenu.add(guidemenu);

JMenuItem whatisnewMI = new JMenuItem("whats New");
whatisnewMI.setMnemonic('e');
guidemenu.add(whatisnewMI);

JMenuItem whatiscoolMI = new JMenuItem("whats cool");
whatiscoolMI.setMnemonic('C');
guidemenu.add(whatiscoolMI);

JMenuItem javaconsolemi = new JMenuItem("java console");
optionsmenu.add(javaconsolemi);

JMenuItem addressbookmi = new JMenuItem("address book");
optionsmenu.add(addressbookmi);
}
```

```
/*
<applet code=TJTable width=500 height=600>
</applet>
*/
import javax.swing.*;
import java.awt.*;
import java.util.*;

public class TJTable extends JApplet
{
    String[] colnames= {"name","size(Bytes)","Date","Directory"};
    Object[][] rowdata = {
        {"autoexec.bat","149","09-11-07",new Boolean(false)},
        {"real","dir","12-11-07",new Boolean(true)},
    };
    public void init()
    {
        JTable table = new JTable(rowdata,colnames);
        getContentPane().add(table.getTableHeader(),BorderLayout.NORTH);
        getContentPane().add(table);
    }
}
```

```
/*
<applet code=TJTree width=500 height=600>
</applet>
*/
import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
import java.awt.event.*;

public class TJTree extends JApplet
{
    Container con;
    public void init()
    {
        con=this.getContentPane();
        DefaultMutableTreeNode root = new DefaultMutableTreeNode("c:\\");
        DefaultMutableTreeNode col11 = new DefaultMutableTreeNode("docs");
        DefaultMutableTreeNode col12 = new
DefaultMutableTreeNode("README");

        root.add(col11);
        root.add(col12);

        DefaultMutableTreeNode col21 = new DefaultMutableTreeNode("API");
        DefaultMutableTreeNode col22 = new
DefaultMutableTreeNode("index.html");
        col11.add(col21);
        col11.add(col22);

        DefaultMutableTreeNode col31 = new
DefaultMutableTreeNode("Swing");
        col21.add(col31);

        DefaultMutableTreeNode col41 = new
DefaultMutableTreeNode("JComponent.html");

```

```
        DefaultMutableTreeNode col42 = new
DefaultMutableTreeNode("JButton.html");
        DefaultMutableTreeNode col43 = new
DefaultMutableTreeNode("JLabel.html");

        col31.add(col41);
        col31.add(col42);
        col31.add(col43);

JTree tree = new JTree(root);

JSScrollPane scrollpane = new JSScrollPane(tree);
con.add(scrollpane);
}
}
```

**randomized** jdk1.5

```
class argstest
{
    public static void main(String[] args)
    {
        int x=Integer.parseInt(args[0]);
        System.out.println("x value is :" +x);

        int y=Integer.parseInt(args[1]);
        System.out.println("x value is :" +y);
    }
}
```

```
class autobox1
{
    static int m(Integer v)
    {
        return v;
    }
    public static void main(String[] args)
    {
        int i=10;
        Integer ival=m(100);
        System.out.println(ival);
    }
}
```

```
class autobox2
{
    public static void main(String[] args)
    {
        Integer ival =100;
        Double dval=98.5;
        dval = dval+ival;
        System.out.println("dval value is!" +dval);
    }
}
```

```
}
```

```
enum Apple
{
    goldendel,reddel,cortland,kashmiri,london
}
class enumdemo
{
    public static void main(String[] args)
    {
        Apple ap;
        ap=Apple.kashmiri;
        System.out.println("value of ap is :" +ap);
        ap=Apple.london;

        if(ap==Apple.london)
            System.out.println("ap contains london ");

        switch(ap)
        {
```

```
        case goldendel: System.out.println("goldendel is yellow");
                                break;

        case london: System.out.println("london delicious is pink");
    }
}
```

```
import java.io.*;
class simple
{
    public static void main(String[] args)
    {
        try
        {
            new class1().meth01();
        }
        catch(exp1 e)
        {
            System.out.println("Cause is:\n" + e.getCause());
            System.out.println("Print StackTrace"+e.printStackTrace());
        }
    }
}//=====//
```

```

//This is a new exception class
class exp1 extends Exception
{
    public exp1() { }
    public exp1(String message)
    {
        super(message);
    }
    public exp1(Throwable throwable)
    {
        super(throwable);
    }
    public exp1(String message, Throwable throwable)
    {
        super(message, throwable);
    }
}//end exp1
//=====================================================================

//This is a new exception class
class exp2 extends Exception
{
    public exp2()
    {
    }
    public exp2(String message)
    {
        super(message);
    }
    public exp2(Throwable throwable)
    {
        super(throwable);
    }
    public exp2(String message, Throwable throwable)
    {
        super(message, throwable);
    }
}//end exp2
//=====================================================================

class Class1
{
    void meth01() throws exp1
    {
        try
        {
            meth02();
        }
    }
}

```

```

        }
    catch(exp2 e)
    {
        System.out.println("Cause is:\n" + e.getCause());
        throw new exp1("Msg from meth01",e);
    }
}
//-----//



void meth02() throws exp2
{
    try
    {
        meth03();
    }
    catch(RuntimeException e)
    {
        System.out.println("Cause is:\n" + e.getCause());
        System.out.println();
        throw new exp2("Msg from meth02",e);
    }//end catch
}//end meth02
//-----//


void meth03()
{
    try
    {
        int x = 3/0;
    }
    catch(ArithmeticException e)
    {
        IndexOutOfBoundsException ex = new IndexOutOfBoundsException("Msg from
meth03");
        ex.initCause(e);
        throw ex;
    }
}

```

Omireddy.Satya



```
import java.io.*;
class simple
{
    public static void main(String[] args)
    {
        try
        {
            new Class1().meth01();
        }
        catch(exp1 e)
        {
            System.out.println("Cause is:\n" + e.getCause());
            // e.printStackTrace();
        }
    }
}//=====

//This is a new exception class
```

```
class exp1 extends Exception
{
    public exp1() { }
    public exp1(String message)
    {
        super(message);
    }
    public exp1(Throwable throwable)
    {
        super(throwable);
    }
    public exp1(String message, Throwable throwable)
    {
        super(message, throwable);
    }
}//end exp1
//=====

//This is a new exception class
class exp2 extends Exception
{
    public exp2()
    {
    }
    public exp2(String message)
    {
        super(message);
    }
    public exp2(Throwable throwable)
    {
        super(throwable);
    }
    public exp2(String message, Throwable throwable)
    {
        super(message, throwable);
    }
}//end exp2
//=====

class Class1
{
    void meth01() throws exp1
    {
        try
        {
            meth02();
        }
    }
}
```

```
        catch(exp2 e)
    {
        System.out.println("Cause is:\n" + e.getCause());
        throw new exp1("Msg from meth01",e);
    }
}
//-----//
```

```
void meth02() throws exp2
{
try
{
meth03();
}
catch(RuntimeException e)
{
System.out.println("Cause is:\n" + e.getCause());
System.out.println();
throw new exp2("Msg from meth02",e);
}//end catch
}//end meth02
//-----//
```

```
void meth03()
{
try
{
int x = 3/0;
}
catch(ArithmeticException e)
{
    IndexOutOfBoundsException ex = new IndexOutOfBoundsException("Msg from
meth03");
    ex.initCause(e);
    throw ex;
}
}
```

ramireddy.Satya

```
import java.util.*;  
  
class formatdemo  
{  
    public static void main(String[] args)  
    {  
        int a=20,b=30;  
        Formatter fmt =new Formatter();  
        fmt.format("formatting %s is easy %d %f ", "with java",10,98.6);  
  
        System.out.printf("default value is %d and %d",a,b);  
  
        System.out.println(fmt);  
    }  
}
```

```

class Gen<T>
{
    T ob;

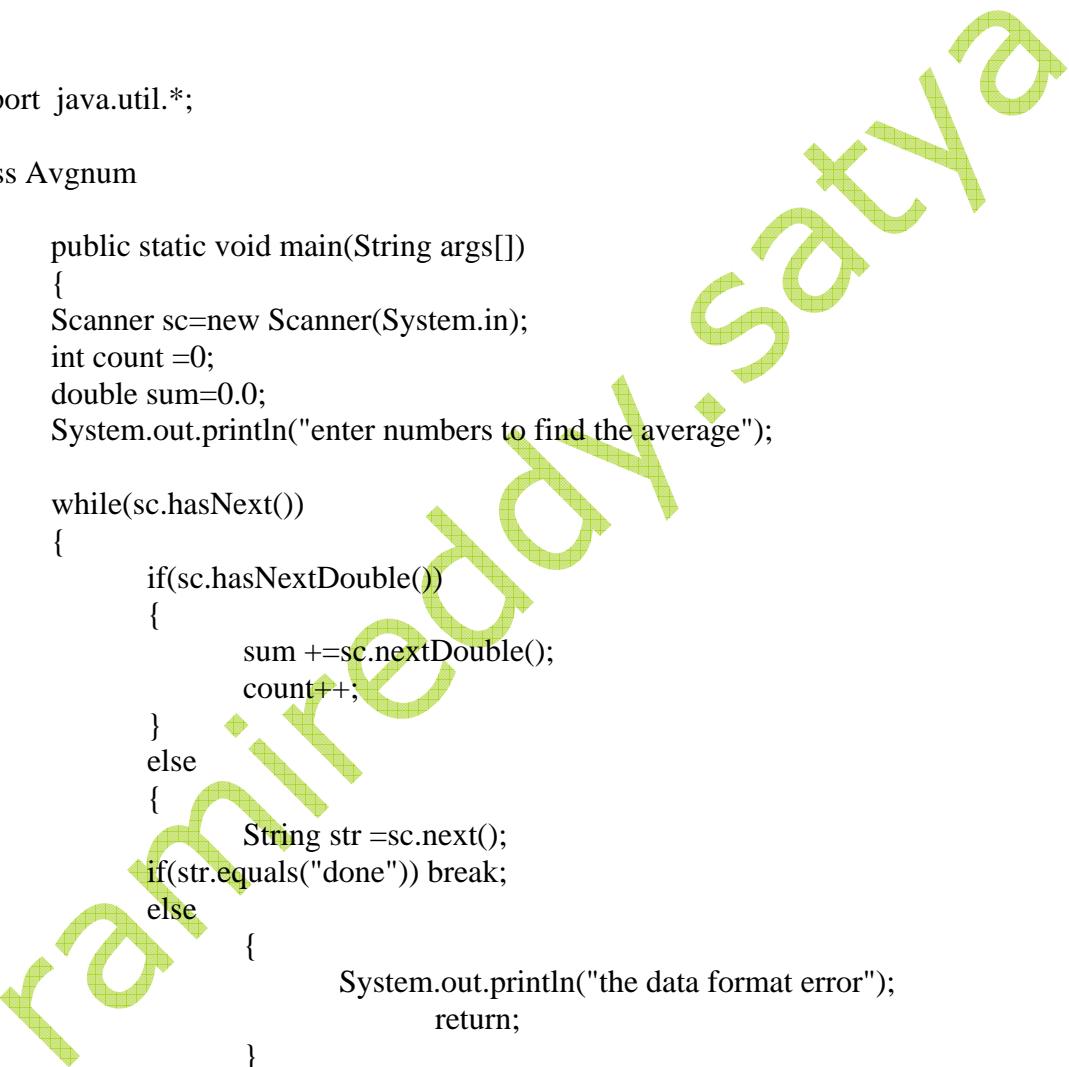
    Gen( T o)
    {
        ob=o;
    }

    T getob()
    {
        return ob;
    }
    void showType()
    {
        System.out.println("The type of T is "+ob.getClass().getName());
    }
}
class Gendemo
{
    public static void main(String[] args)
    {
        Gen<Integer> iob=new Gen<Integer>(66);
        iob.showType();

        int v= iob.getob();
        System.out.println("value is :" +v);

        Gen<String> strob = new Gen<String>("generics test");
        strob.showType();
        String str = strob.getob();
        System.out.println("value is "+str);
    }
}

```



```
import java.util.*;

class Avgnum
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int count =0;
        double sum=0.0;
        System.out.println("enter numbers to find the average");

        while(sc.hasNext())
        {
            if(sc.hasNextDouble())
            {
                sum +=sc.nextDouble();
                count++;
            }
            else
            {
                String str =sc.next();
                if(str.equals("done")) break;
                else
                {
                    System.out.println("the data format error");
                    return;
                }
            }
        }
        System.out.println("the average is :" +sum/count);
    }
}
```

```
import static java.lang.System.*;
class Test
{
    public static void main(String[] args)
    {
        out.println("Hello World!");
        out.println("hello this is core class");
        out.println("how r u ");
        out.println("hyd");
    }
}
```

```
class passarray
{
    static void vatest(int ... v)
    {
        System.out.print("number of args:"+v.length+"contents");

        for(int x : v)
            System.out.print(x+" ");
            System.out.println();
    }
    public static void main(String[] args)
    {
        //int n1[ ] = {10};
        //int n2[ ] = {1,2,3};
        //int n3[ ] = {};
        vatest(10);
        vatest(10,20);
        vatest();
    }
}
```

```
class passarray2
{
    static void vatest(int ... v)
    {
        System.out.print("number of args:"+v.length+"contents");

        for(int x : v)
            System.out.print(x+" ");
            System.out.println();
    }
    public static void main(String[] args)
    {
        vatest(10);
        vatest(1,2,3);
        vatest();
    }
}
```

**Komireddy.Satya**

sliptest

```
class NFE
{
    public static void main(String[] args)
    {
        String s="22";
        try
        {
            s=s.concat(".5");
            double d = Double.parseDouble(s);
            s=Double.toString(d);
            int x = (int)(Double.valueOf(s).doubleValue());
            System.out.println(x);
        }
    }
}
```

```
        catch(NumberFormatException e)
        {
            System.out.println("bad number");
        }
    }

/*
what is the result?
A. 22
B. 22.5
C. 23
D. bad number
E. Compilation fails
F. An uncaught exception is thrown at runtime.
*/
```

```
class C extends A
{
    int getNum()
    {
        return num;
    }
}

class B
{
    public int name;
}

class A
{
    B b = new B();
```

```
    public int num;  
}
```

which two are true about instances of the classes listed above?(Choose two)

- A. B is - a A
- B. C is - a A
- C. A has - a C
- D. C has - a B
- E. B has - a A

```
class Over  
{  
    int doStuff(int a, float b)  
    {  
        return 7;  
    }  
}  
class Over2 extends Over  
{  
    // insert code here  
}  
  
/*
```

which two methods if inserted independently at line 10 will compile(choose two)

- A. private int doStuff(int x, float y) { return 4; }
- B. protected int doStuff(int x, float y) { return 4; }

C. Integer doStuff(int d,float e) { return 4;}  
D. long doStuff(int x,float y) { return 4;}  
E. int doStuff(float x,int y) { return 4;}  
\*/

//Given the following.

```
class A
{
    public void m1()
    {
        System.out.print("ma");
    }
}
class B extends A
{
    public static void main(String args[])
    {
```

```
A a = new B();
a.m1();
}
public void m1()
{
    System.out.print("mb");
}
}

/*
what is the result?
A. ma
B. mb
C. mamb
D. Compilation fails
E. An exception is thrown at runtime
*/
```

```
class test
{
    public static void main(String args[])
    {
        char ch;
        String test2 = "abcd";
        String test = new String("abcd");
        if(test.equals(test2))
        {
            if(test == test2)
                ch=test.charAt(0);
            else
                ch=test.charAt(1);
```

```
        }
    else
    {
        if(test == test2)
            ch=test.charAt(2);
        else
            ch=test.charAt(3);
    }
    System.out.println(ch);
}
}
```

select correct answer

- A. 'a'
- B. 'b'
- C. 'c'
- D. 'd'

//Given the following

```
class CommandArgs
{
    public static void main(String args[])
    {
        //String [] argh;
        int x;
        x=args.length;
        for(int y=0;y<=x;y++)
        {
            System.out.print(args[y]);
        }
    }
}
```

```
        }
    }

/*
and the command-line invocation
    java CommandArgs 1 2 3

what is the result?
A. 0 1 2
B. 1 2 3
C. 0 0 0
D. null null null
E. Compilation fails
F. 1 2 3 then an exception is thrown at
runtime
*/
```

Given the following.

```
public class Test
{
    public static void main(String args[])
    {
        boolean b1 = true;
        boolean b2 = false;
        boolean b3 = true;
        if(b1 & b2 | b2 | b3 & b2)
            System.out.print("hai");
        if(b1 & b2 | b2 & b3 | b2 | b1)
            System.out.println("hello");
```

```
    }  
}
```

what is the result?

- A. hai
- B. hello
- C. hai hello
- D. no output is produced
- E. Compilation error
- F. An exception is thrown at runtime

//Given the following.

```
class TestStr  
{  
    public static void main(String args[])  
    {  
        String s1 ="1";  
        String s2 = "2";  
        String s3 = s1;  
        s1="3";  
        System.out.println(s1 + s2 + s3);  
    }  
}
```

```
/*
what is the result?
A. 6
B. 123
C. 321
D. 323
E. compilation fails.
F. An exception is thrown at runtime.
*/
```

```
class test
{
    public static void main(String args[])
    {
        int [] arr = {1,2,3,4};
        call_array(arr[0],arr);
        System.out.println(arr[0]+ " , "+arr[1]);
    }
    static void call_array(int i,int arr[])
    {
        arr[i]=6;
        i=5;
    }
}
```

}

- A. 1,2
- B. 5,2
- C. 1,6
- D. 5,6

```
public class RTEExcept
{
    public static void throwit()
    {
        System.out.print("throw it");
        throw new RuntimeException();
    }
    public static void main(String args[])
    {
        try
        {
            System.out.print("hello");
            throwit();
        }
        catch(Exception e)
        {
            System.out.print("caught");
        }
        finally
        {
            System.out.print("finally");
        }
        System.out.println("after");
    }
}
```

what is the result?

- A. hello throwit caught
- B. Compilation fails
- C. hello throwit RuntimeException caught after
- D. hello throwit RuntimeException
- E. hello throwit caught finally after
- F. hello throwit caught finally after  
RuntimeException

Given the following.

```
class LoopTest
{
    public static void main(String args[])
    {
        int a =1;
        while(a++<=1)
            while(a++<=2)
                while(a++<=3);
        System.out.println("a value is :" +a);
    }
}
```

what is the result

- A.10
- B.8
- C.6
- D.7
- E.None

//Given the following.

```
class TestDogs
{
    public static void main(String args[])
    {
        Dog [][] [] theDogs = new Dog[3][2][];
        System.out.println(theDogs[2][0][0].toString());
    }
}
class Dog{}  
/*
```

what is the result

- A. null
- B. the dogs
- C. Compilation fails
- D. an exception is thrown at run time;

//what gets printed when the following code is compiled and run with the following command

```
//java test 2

class test
{
    public static void main(String args[])
    {
        Integer intObj = Integer.valueOf(args[args.length-1]);
        int i = intObj.intValue();
        if(args.length >1)
            System.out.println(i);
        if(args.length >0)
            System.out.println(i-1);
        else
            System.out.println(i-2);
    }
}
/*
```

Select the correct answer

- A. test
- B. test -1
- C. 0
- D. 1
- E. 2

ramireddy.Satya