**JAVA SHORT QUESTIONS AND ANSWERS**

**1.What is the significance of each word in public static void main(String args[])?**

Ans: "**public**" means that main() can be called from anywhere.  
"**static**" means that main() doesn't belong to a specific object  
"**void**" means that main() returns no value  
"**main**" is the name of a function. main() is special because it is the start of the program.  
"**String[]**" means an array of String.  
"**args**" is the name of the String[] (within the body of main()). "args" is not special; you could name it anything else and the program would work the same.

**2. What is the use of super keyword?**

#### Ans: 1) super.<variable\_name> refers to the variable of variable of parent class. 2) super() invokes the constructor of immediate parent class. 3) super.<method\_name> refers to the method of parent class.

Example:

class Parentclass

{

int num=100;

}

//Child class or subclass

class Subclass extends Parentclass

{

int num=110;

void printNumber(){

//Super.variable\_name

System.out.println(super.num);

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printNumber();

}

}

Output: 100

1. **What is the normal priority of a thread and how the priority of a thread can be changed?**

Ans: JVM selects to run a Runnable thread with the highest priority.

All Java threads have a priority in the range 1-10.

Top priority is 10, lowest priority is 1.Normal

priority ie. priority by default is 5.

Thread.MIN\_PRIORITY - minimum thread priority

Thread.MAX\_PRIORITY - maximum thread priority

Thread.NORM\_PRIORITY - maximum thread priority

Whenever a new Java thread is created it has the same priority as the thread which created it.

Thread priority can be changed by the setpriority() method.

1. **List different collection classes and collection interfaces?**

|  |  |
| --- | --- |
| [Java Collections Classes](http://www.journaldev.com/1260/collections-in-java-tutorial#java-collections-classes)   * [HashSet Class](http://www.journaldev.com/1260/collections-in-java-tutorial#hashset-class) * [TreeSet Class](http://www.journaldev.com/1260/collections-in-java-tutorial#treeset-class) * [ArrayList Class](http://www.journaldev.com/1260/collections-in-java-tutorial#arraylist-class) * [LinkedList Class](http://www.journaldev.com/1260/collections-in-java-tutorial#linkedlist-class) * [HashMap Class](http://www.journaldev.com/1260/collections-in-java-tutorial#hashmap-class) * [TreeMap Class](http://www.journaldev.com/1260/collections-in-java-tutorial#treemap-class) * [PriorityQueue Class](http://www.journaldev.com/1260/collections-in-java-tutorial#priority-queue) | [Java Collections Interfaces](http://www.journaldev.com/1260/collections-in-java-tutorial#java-collections-interfaces)   * [Collection Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#collection-interface) * [Iterator Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#iterator-interface) * [Set Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#set-interface) * [List Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#list-interface) * [Queue Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#queue-interface) * [Dequeue Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#dequeue-interface) * [Map Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#map-interface) * [ListIterator Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#listiterator-interface) * [SortedSet Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#sortedset-interface) * [SortedMap Interface](http://www.journaldev.com/1260/collections-in-java-tutorial#sortedmap-interface) |

1. **What is the use of string Tokenizer?**

Ans: The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string.

It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc. like StreamTokenizer class.

The **tokenization method** is much simpler than the one used by the StreamTokenizer class. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments

Example:

**import** java.util.StringTokenizer;

**public** **class** Simple{

**public** **static** **void** main(String args[]){

   StringTokenizer st = **new** StringTokenizer("my name is khan"," ");

**while** (st.hasMoreTokens()) {

         System.out.println(st.nextToken());

     }

   }

}

Output:my

name

is

khan

1. **Explain the delegation event model?**

Ans: The event model is based on the Event Source and Event Listeners. Event Listener is an object that receives the messages / events. The Event Source is any object which creates the message / event. The Event Delegation model is based on – The Event Classes, The Event Listeners, Event Objects.  
  
There are three participants in event delegation model in Java;  
  
- Event Source – the class which broadcasts the events  
- Event Listeners – the classes which receive notifications of events  
- Event Object – the class object which describes the event.  
  
An event occurs (like mouse click, key press, etc) which is followed by the event is broadcasted by the event source by invoking an agreed method on all event listeners. The event object is passed as argument to the agreed-upon method. Later the event listeners respond as they fit, like submit a form, displaying a message / alert etc.

1. **List the different AWT controls?**

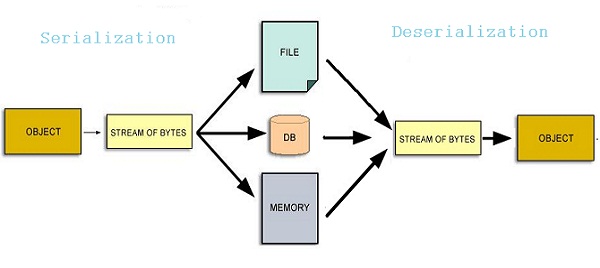
Ans: - Controls are components that allow a user to interact with your application.  
- A component is a graphical object.   
- A few examples of components are:  
1. Button  
2. Canvas  
3. Checkbox  
4. Choice  
5. Container  
6. Label  
7. List  
8. Scrollbar  
9. TextComponent

1. **what is serialization? Which type of variables cannot be serialized?**

**Ans:** To **serialize** an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object. A **Java** object is **serializable** if its class or any of its superclasses implements either the **java**.io.**Serializable** interface or its subinterface, **java**.io.Externalizable.

java serialization

Static and transcient variables cannot be serialized



1. **What is the use of data output stream in java and push back reader?**

Ans: The **Java**.io.**DataOutputStream** class lets an **application** write primitive **Java data**types to an **output stream** in a portable way. An **application** can then **use** a **data** input**stream** to read the **data** back in.

The PushbackReader class allows one or more characters to be returned to the input stream. This allows you to peek in the input stream.

Here are its two constructors:

* PushbackReader(Reader inputStream)   
  creates a buffered stream that allows one character to be pushed back.
* PushbackReader(Reader inputStream, int bufSize)   
  the size of the pushback buffer is passed in bufSize.

1. **Differences between overriding and overloading?**

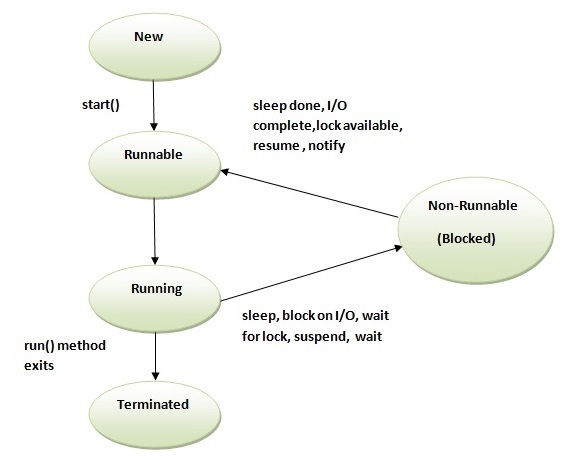


|  |  |
| --- | --- |
| Overriding  1) Methods name and signatures must be same.  2)Overriding is the concept of runtime polymorphism  3)When a function of base class is re-defined in the derived class called as Overriding  4)It needs inheritance.  5)Method should have same data type.  6)Method should be public. | Overloading  1) Having same method name with different Signatures.  2)Overloading is the concept of compile time polymorphism  3)Two functions having same name and return type, but with different type and/or number of arguments is called as Overloading  4)It doesn't need inheritance.  5)Method can have different data types  6)Methods can be different access specifies. |

1. **what are the different states of a thread?**

**A thread can be in one of the following states:**

* NEW. A thread that has not yet started is in this state.
* RUNNABLE. A thread executing in the Java virtual machine is in this state.
* BLOCKED. A thread that is blocked waiting for a monitor lock is in this state.
* WAITING. ...
* TIMED\_WAITING. ...
* TERMINATED.



1. **What is the result of java expression 17.0/0.0?**

Ans:  A divide by zero error generates a processor exception which triggers an interrupt. The interrupt is "read" by the operating system and forwarded to the program if a handler is registered. Since Java registers a handler, it receives the error and then translates it into an ArithmeticException that travels up the stack.

1. **Which is the base class for all events?**

Ans: The base class, from which all events inherit, is java.util.EventObject. **AWT** defines its own base class for GUI events, java.**awt**.**AWTEvent**, which is subclassed from EventObject. **AWT** then defines a number of subclasses of **AWTEvent** in the package java.**awt**.event.

1. **Differences between a component and a container?**

### Containers

Containers hold and organize your Components, but they also contain code for event handling and many 'niceties' such as controlling the cursor image and the application's icon.

**Containers:**

* Frame
* Window
* Dialog
* Panel

### Component

Components are generally the stuff that the user interacts with your application.

List of common Components

* List
* Scrollbar
* TextArea
* TextField
* Choice
* Button
* Label

1. **How many catch clauses can a try/catch statement contain?**

Ans: 1. A try block can have any number of catch blocks.  
2. A catch block that is written for catching the class Exception can catch all other exceptions  
Syntax:

catch(Exception e){

  //This catch block catches all the exceptions

}

3. If multiple catch blocks are present in a program then the above mentioned catch block should be placed at the last as per the exception handling best practices.  
4. If the try block is not [**throwing any exception**](http://beginnersbook.com/2013/04/throw-in-java/), the catch block will be completely ignored and the program continues.

5. If the try block [**throws an exception**](http://beginnersbook.com/2013/04/java-throws/), the appropriate catch block (if one exists) will catch it  
–catch(ArithmeticException e) is a catch block that can catch ArithmeticException  
–catch(NullPointerException e) is a catch block that can catch NullPointerException  
6. All the statements in the catch block will be executed and then the program continues.

Example:

class Example2{

public static void main(String args[]){

try{

int a[]=new int[7];

a[4]=30/0;

System.out.println("First print statement in try block");

}

catch(ArithmeticException e){

System.out.println("Warning: ArithmeticException");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Warning: ArrayIndexOutOfBoundsException");

}

catch(Exception e){

System.out.println("Warning: Some Other exception");

}

System.out.println("Out of try-catch block...");

}

}

1. **Difference between iterator and list-iterator?**

There are two **differences**: We can use **Iterator** to traverse Set and List and also Map type of Objects. While a**ListIterator** can be used to traverse for List-type Objects, but not for Set-type of Objects. ... By using **Iterator** we can retrieve the elements from Collection Object in forward direction only

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | List-Iterator | Iterator |
| Traversal Direction | | Both , forward and backward | Forward |
| Objects traversal | | List only | Map, Set and List |
| Add and Set operations | | Allows both operations | Not possible |
| Iterator's current position | | Yes , can be determined | Not possible. |
| Retrieve Index | | Yes | Not possible |

1. **What is the purpose of any java.io class that contains word buffer in its class?**

Ans: Package **java**.**io**. Provides for system input and output through data streams, serialization and the file system. A Closeable is a source or destination of data that can be closed. ... Serializability of a **class** is enabled by the **class** implementing the**java**.**io**.Serializable interface.

1. **What are the differences between start() and run() in java threads?**

Ans: The **difference** is that **Thread**.**start**() **starts** a **thread**, while Runnable.**run**() just calls a method. Actually **Thread**.**start**() creates a new **thread** and have its own execution scenario. ... Another **difference between start** vs **run in Java thread** is that you can not call **start**() method twice on **thread** object.

Examples:

1.run()

Thread one = new Thread();  
Thread two = new Thread();  
one.run();  
two.run();

2.start()

Thread one = new Thread();   
Thread two = new Thread();  
one.start();  
two.start();

1. **Write differences between public and default access levels of a member of a class**

The basic Accessibility Modifiers are of 4 types in Java. They are

1. public
2. protected
3. package/default
4. private

* Public:  If top level class or interface within a package is declared as Public, then it is

accessible both inside and outside of the package.

* Default**:** If no access modifier is specified in the declaration of the top level class or

interface, then it is accessible only within package level. It is not accessible in

other packages or sub packages.

1. **What are the differences between capacity and size of a vector class?**

|  |  |  |
| --- | --- | --- |
| Return | Method | Explanation |
| int | capacity() | Returns the current capacity of this vector. |
| Int | size() | Returns the number of components in this vector. |

After capacity, we can set size and we can get size() of a vector class.

1. **Difference between function and method? Define instance method?**

|  |  |  |
| --- | --- | --- |
| **Sno** | **Functions** | **Methods** |
| **1** | **Functions do not have any reference variables** | **Methods are called b reference variables** |
| **2** | **All data that is passed to a function is explicitly passed** | **It is implicitly passed the object for which it was called** |
| **3** | **It does not have access controlling** | **It has access controlling** |
| **4** | **Function applies to both object oriented and non-object oriented language** | **Method applies to only object oriented programming languages.** |
| **5** | **Eg: javascript** | **Eg: c++, java** |

A instance method is associated with and operates upon an object. Therefore, it is necessary to create an instance of that class in order to invoke such a method.

### Example

class InstanceMethod

{

public static void main(String [] args){

InstanceMethod obj = new InstanceMethod();// because that method we wrote is instance we will write an object to call it

System.out.println(obj.sum(3,2));

}

int f;

public double sum(int x,int y){// this method is instance method

f = x+y;

return f;

}

}

1. **What is hashcode? How do you find hash code of an object?**

Ans: The **hashcode** of a **Java** Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure. **hash code** is an unique id number allocated to an object by JVM. If two objects are equals then these two objects should return same **hash code**.

Finding hash code of an object:

Basically the default implementation of **hashCode**() provided by **Object** is derived by mapping the memory address to an integer value. If look into the source of **Object** class , you will find the following **code** for the **hashCode**

1. **what is the scope of default access specifier in java? How it is different from protected access specifier?**

Ans: **Java** provides a **default specifier** which is used when no **access** modifier is present. Any class, field, method or constructor that has no declared **access** modifier is accessible only by classes in the same package. The **default** modifier is not used for fields and methods within an interface.

Difference:

 A member with no **access modifier** is only accessible within classes in the same **package**. A **protected** member is accessible within all classes in the same**package** and within subclasses in **other** packages.

1. **what is the difference between throw and throws clause in java?**

|  |  |  |
| --- | --- | --- |
| sno | throw | Throws |
| 1 | Java throw keyword is used to explicitly throw an exception | Java throws keyword is used to declare an exception |
| 2 | Checked exception cannot be propagated using throw only. | Checked exception can be propagated with throws. |
| 3 | Throw is followed by an instance. | Throws is followed by class. |
| 4 | Throw is used within the method. | Throws is used with the method signature. |
| 5 | You cannot throw multiple exceptions | You can declare multiple exceptions |
| 6 | **void** m(){  **throw** **new** ArithmeticException("sorry");  } | Public void method()throws IOException,SQLException |

1. **What is string constant pool?**

Ans:  It is a special place where the collection of references to string objects are placed.

public class StringConstantPool {

public static void main(String[] args) {

String s = "prasad";

String s2 = "prasad";

System.out.println(s.equals(s2));

System.out.println(s == s2);

}

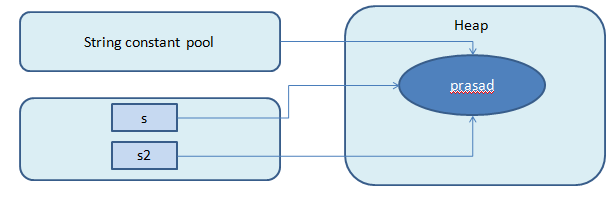
}

Output: True

True

In below diagram

* The class is loaded when JVM is invoked.
* JVM will look for all the string literals in the program
* First, it finds the variable s which refers to the  literal “prasad” and it will be created in the memory
* A reference for the literal “prasad” will be placed in the string constant pool memory.
* Then it finds another variable s2 which is referring to the same string literal “prasad“.
* Now that JVM has already found a string literal “prasad“, both the variables sand s2 wil refer to the same object i.e. “prasad“.



public class StringConstantPool {

public static void main(String[] args) {

String s = "prasad";

String s2 = new String("prasad");

System.out.println(s.equals(s2));

System.out.println(s == s2);

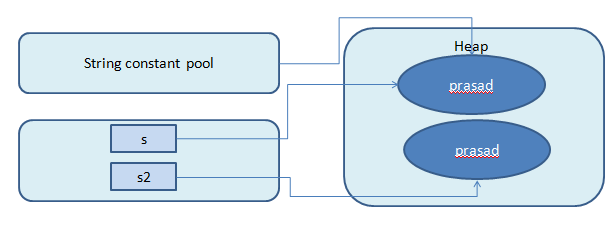
}

}

Output

True

False

* The contents of both objects are the same so equals method returns true
* The objects referred by both variables are different so == operator returns false

1. **Does collection object stores copies of other objects or their references in java? what is a collection framework?**

The collection object stores references to the objects, not a copy.

The **Java collections framework** (JCF) is a set of classes and interfaces that implement commonly reusable **collection** data structures. Although referred to as a**framework**, it works in a manner of a library. The JCF provides both interfaces that define various **collections** and classes that implement them.

1. **State the default priority of a thread? What is a daemon thread?**

JVM selects to run a Runnable thread with the highest priority.

All Java threads have a priority in the range 1-10.

Top priority is 10, lowest priority is 1.

**Normal priority ie. priority by default is 5.**

A **daemon thread** is a **thread** that does not prevent the JVM from exiting when the program finishes but the **thread** is still running. An example for a **daemon thread** is the garbage collection. You can use the setDaemon(boolean) method to change the**Thread daemon** properties before the **thread** starts

1. **Explain event handling mechanism?**

Any action that user performs on a GUI component must be listened and necessary action should to be taken. For example, if a user clicks on a *Exit* button, then we need to write code to exit the program. So for this, we need to know that the user has clicked the button. This process of knowing is called as listening and the action done by the user is called an event. Writing the corresponding code for a user action is called as **Event handling.**

The java.awt.event package provides many event classes and Listener interfaces for event handling.

|  |  |
| --- | --- |
| **Event Classes** | **Listener Interfaces** |
| ActionEvent | ActionListener |
| MouseEvent | MouseListener and MouseMotionListener |
| MouseWheelEvent | MouseWheelListener |
| KeyEvent | KeyListener |
| ItemEvent | ItemListener |
| TextEvent | TextListener |
| AdjustmentEvent | AdjustmentListener |
| WindowEvent | WindowListener |
| ComponentEvent | ComponentListener |
| ContainerEvent | ContainerListener |
| FocusEvent | FocusListener |

Following steps are required to perform event handling:

1. Implement the Listener interface and overrides its methods
2. Register the component with the Listener

Sample code:

public class ButtonDemo extends Frame implements ActionListener

{

  public ButtonDemo()

  {

    Button btn = new Button("OK");

    btn.addActionListener(this);

    add(btn);

  }

  public void actionPerformed(ActionEvent e)

  {

    String str = e.getActionCommand();

  }

}

1. List different layouts?

* BorderLayout.
* BoxLayout.
* CardLayout.
* FlowLayout.
* GridBagLayout.
* GridLayout.
* GroupLayout.
* SpringLayout

1. **Is it possible to compile and run the java program without main method?**

Yes You can **compile and execute without main method** By using static block. But after static block executed (printed) you will get an error saying no **main method** found.

1. **What is the difference between #include and import?**

#include directive makes the compiler go to the C/C++ standard library and copy the code from the header files into the program. As a result, the program size increases, thus wasting memory and processor’s time.   
import statement makes the JVM go to the Java standard library, execute the code there , and substitute the result into the program. Here, no code is copied and hence no waste of memory or processor’s time.hence import is an efficient mechanism than #include.

1. **Define terms widening and narrowing?**

For a primitive data types, a value narrower data type can be converted to a value of a broader data type without loss of information. This is called Widening conversion. For example an int is directly converted to a double without first having to convert it to a long and a float.  
  
Converting from a broader data type to a narrower data type is called narrowing conversion, which can result in loss of information. For example conversion between char and byte and short are narrowing conversions. For example byte takes 8 bits, short takes 16 bits, int takes 32 bits, long takes 64 bits. Conversion of short to int or int to long is widening conversion and conversion of int to short is a narrowing conversion

Java's widening conversions are

From a **byte** to a **short**, an **int**, a **long**, a **float**, or a **double**

From a **short** to an **int**, a **long**, a **float**, or a **double**

From a **char** to an **int**, a **long**, a **float**, or a **double**

From an **int** to a **long**, a **float**, or a **double**

From a **long** to a **float** or a **double**

From a **float** to a **double**

Narrow conversions

From a **byte** to a **char**

From a **short** to a **byte** or a **char**

From a **char** to a **byte** or a **short**

From an **int** to a **byte**, a **short**, or a **char**

From a **long** to a **byte**, a **short**, a **char**, or an **int**

From a **float** to a **byte**, a **short**, a **char**, an **int**, or a **long**

From a **double** to a **byte**, a **short**, a **char**, an **int**, a **long**, or a **float**

1. **why methods in interface are public and abstract by default? Can you implement from interface from another?**

Java **interface methods** aren't only **public** by default - they can only be **public**. The reason for this is because an **interface method** is a specification meant for consumption by the **public** (in Java terms - meaning, in any class). The **interface method** enforces that the implementing class **method** is **public**.

**interfaces** have no **implementation** so that's not possible. An **interface can**however extend **another interface**, which means it **can** add more methods and inherit its type.

1. **List out advantages of stream concept? What is the default buffer size for any buffered class?**

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination. InputStream and OutputStream are the basic stream classes in Java.

All the other streams just add capabilities to the basics, like the ability to read a whole chunk of data at once for performance reasons (BufferedInputStream) or convert from one kind of character set to Java's native unicode (Reader), or say where the data is coming from (FileInputStream, SocketInputStream and ByteArrayInputStream, etc.)

A FileInputStream is an InputStream to obtain bytes from a file. It is used for reading inputs of raw bytes such as image data. It can read byte oriented data.

The **default buffer size** of 8192 chars

1. **Can you synchronize arraylist object?**

ArrayList is non-synchronized and should not be used in multi-thread environment without explicit synchronization.

**There are two ways to synchronize explicitly:**

1. Using Collections.synchronizedList() method
2. Using thread-safe variant of ArrayList: CopyOnWriteArrayList
3. **Difference between extends thread between implements runnable ?**

Thread is a block of code which can execute concurrently with other threads in the JVM. You can create and run a thread in either ways; Extending **Thread** class, Implementing **Runnable**interface.

Both approaches do the same job but there have been some differences. The most common difference is

* When you **extends Thread** class, after that you can’t extend any other class which you required. (As you know, Java does not allow inheriting more than one class).
* When you **implements Runnable**, you can save a space for your class to extend any other class in future or now.

However, the significant difference is.

* When you **extends Thread** class, each of your thread creates unique object and associate with it.
* When you **implements Runnable**, it shares the same object to multiple threads.

1. **why private methods and final methods are same justify**

A private method is automatically final and hidden from its derived class. A final class is not hidden from its derived class. Therefore you can make a new class with the same name as the private method such as

class test {

private void works {

}

}

class tester extends test {

private void works {

}

}

but you cannot make a new class with the same name as the final method

/\*----------Doesn't Work------------\*/

class test {

final void dWorks {

}

}

class tester extends test {

final void dWorks {

}

}

1. What is a frame?

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

EXAMPLE:

import java.awt.\*;  
class AWTFrame extends Frame  
{  
    public AWTFrame()  
    {  
    setTitle("AWT Frame"); // Set the title  
    setSize(400,400); // Set size to the frame  
    setVisible(true); // Make the frame visible  
    setBackground(Color.red); // Set the background  
    setExtendedState(MAXIMIZED\_BOTH);//Make the frame maximized  
    setCursor(Cursor.HAND\_CURSOR); // Deprecated  
    }  
      
    public static void main(String args[])  
    {  
    new AWTFrame();  
    }  
}

1. Write a note on applet?

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

### Advantage of Applet

There are many advantages of applet. They are as follows:

* It works at client side so less response time.
* Secured
* It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

### Drawback of Applet

* Plugin is required at client browser to execute applet.

1. Benfits of object oriented programming?

Code Reuse and Recycling: Objects created for Object Oriented Programs can easily be reused in other programs.

Encapsulation : Once an Object is created, knowledge of its implementation is not necessary for its use. In older programs, coders needed understand the details of a piece of code before using it (in this or another program).

Design Benefits: Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with less flaws. In addition, once a program reaches a certain size, Object Oriented Programs are actually easier to program than non-Object Oriented ones.

Software Maintenance: Programs are not disposable. Legacy code must be dealt with on a daily basis, either to be improved upon (for a new version of an exist piece of software) or made to work with newer computers and software. An Object Oriented Program is much easier to modify and maintain than a non-Object Oriented Program. So although a lot of work is spent before the program is written, less work is needed to maintain it over time.

1. what are iterators ?

*Java iterator* is an interface belongs to collection framework allow us to traverse the collection and access the data element of collection without bothering the user about specific implementation of that collection it. Basically List and set collection provides the iterator. You can get Iterator from ArrayList, LinkedList, and TreeSet etc.

1. what are comparators?

Comparators are used to compare objects.

obj1 and obj2 are the objects to be compared. comparator method returns zero if the objects are equal. It returns a positive value if obj1 is greater than obj2. Otherwise, a negative value is returned.

By overriding compare( ), you can alter the way that objects are ordered. For example, to sort in a reverse order, you can create a comparator that reverses the outcome of a comparison.