Nexwave
Java – Practise - Mock Questions & Answers

Contents

Language Fundamentals Mock Questions

1.1) Which of the following lines are valid declarations?
(a) char a = '\u0061';
(b) char \u0061 = 'a';
(c) ch\u0061r a = 'a';
(d) char a = (char) 65;

1.2) Is an empty file a valid source file?
(a) True.
(b) False.

1.3) Which of these are valid declaration of the main() method?
(a) static void main(String args[]) {/* ... */}
(b) public static int main(String args[]) {/* ... */}
(c) public static void main(String args) {/* ... */}
(d) final static public void main(String[] arguments) {/* ... */}
(e) public int main(String args[], int argc) {/* ... */}
(f) public void main(String args[]) {/* ... */}

1.4) Which one of the following are not valid character contants?
Select any two.
(a) char c = '\u00001' ;
(b) char c = '\101';
(c) char c = 65;
(d) char c = '\1001' ;

1.5) You have the following code in a file called Test.java

```
class Base{
   public static void main(String[] args){
   System.out.println("Hello");
   }
}

public class Test extends Base{}
```

What will happen if you try to compile and run this?
(a) It will fail to compile.
(b) Runtime error
(c) Compiles and runs with no output.
(d) Compiles and runs printing "Hello"

1.6) What is the result of trying to compile and run the following code.

```
public final static void main(String[] args){
double d = 10.0 / -0;
 if(d == Double.POSITIVE_INFINITY)
  System.out.println("Positive infinity");
  else
  System.out.println("Negative infinity");
}
```

(a) output Positive infinity
(b) output Negative infinity
(c) Will fail to compile
(d) Runtime exception

1.7) What is the result that will be printed out ?

```
void aMethod() {
      float f = (1 / 4) * 10;
      int i = Math.round(f);
      System.out.println(i);
```

```
        }
```

(a) 2
(b) 0
(c) 3
(d) 2.5
(e) 25

1.8) Which of the following are valid declarations?
Note : None of the literals used here contain the character O they are all zeroes.
(a) 1. int i = 0XCAFE;
(b) 2. boolean b = 0;
(c) 3. char c = 'A';
(d) 4. byte b = 128;
(e) 5. char c = "A";

1.9) What is the result of trying to compile and run this program.

```
public class Test{
   public static void main(String[] args){
    int[] a = {1};
    Test t = new Test();
    t.increment(a);
    System.out.println(a[a.length - 1]);
   }
   void increment(int[] i){
   i[i.length - 1]++;
   }
}
```

(a) Compiler error.
(b) Compiles and runs printing out 2
(c) Compiles and runs printing out 1
(d) An ArrayIndexOutOfBounds Exception at runtime

1.10) What will happen if you try to compile and run this ?

```
public class Test{
   static{
    print(10);
    }
  static void print(int x){
    System.out.println(x);
    System.exit(0);
   }
}
```

(a) Compiler error.
(b) Will throw a NoSuchMethod error at runtime.
(c) It will compile and run printing out "10"
(d) It will run with no output.
(e) It will run and print "10" and then crash with an error.

1.11) Is this legal?

```
long longArr[];
int intArr[] = { 7 ,8 , 9};
longArr = intArr;
```

(a) Yes
(b) No

1.12) The range of a byte is from -127 to 128
(a) True
(b) False

1.13) Identify the valid assignments.
(a) float f = \u0038;

(b) long L2 = 2L;
(c) float f = 1.2;
(d) char c = '/u004E';
(e) byte b = 100;

1.14) What is the result of trying to compile and run the following code.

```
public static void main(String[] args){
double d = 10 / 0;
if(d == Double.POSITIVE_INFINITY)
  System.out.println("Positive infinity");
else
  System.out.println("Negative infinity");
}
```

(a) output Positive infinity
(b) output Negative infinity
(c) Will fail to compile
(d) Runtime exception

1.15) Which statement is true about a method? Select the one correct answer.
(a) A method is an implementation of an abstraction.
(b) A method is an attribute defining the property of a particular abstraction.
(c) A method is a category of objects.
(d) A method is an operation defining the behavior for a particular abstraction.
(e) A method is a blueprint for making operations.

1.16) Which statement is true about an object? Select the one correct answer.
(a) a. An object is what classes are instantiated from.
(b) b. An object is an instance of a class.
(c) c. An object is a blueprint for creating concrete realization of abstractions.
(d) d. An object is a reference to an attribute.
(e) e. An object is a variable.

1.17) Which line contains a constructor in this class definition?

```
public class Counter { // (1)
int current, step;
public Counter(int startValue, int stepValue) { // (2)
set(startValue);
setStepValue(stepValue);
}
public int get() { return current; } // (3)
public void set(int value) { current = value; } // (4)
public void setStepValue(int stepValue) { step = stepValue; } // (5)
}
```

Select the one correct answer.
(a) Code marked with (1) is a constructor.
(b) Code marked with (2) is a constructor.
(c) Code marked with (3) is a constructor.
(d) Code marked with (4) is a constructor.
(e) Code marked with (5) is a constructor.

1.18) Given that Thing is a class, how many objects and how many reference variables are created by the following code?

```
Thing item, stuff;
item = new Thing();
Thing entity = new Thing();
```

Select the two correct answers.
(a) One object is created.
(b) Two objects are created.
(c) Three objects are created.
(d) One reference variable is created.
(e) Two reference variables are created.

(f) Three reference variables are created.

1.19) Which statement is true about an instance method? Select the one correct answer.
(a) An instance member is also called a static member.
(b) An instance member is always a field.
(c) An instance member is never a method.
(d) An instance member belongs to an instance, not to the class as a whole.
(e) An instance member always represents an operation.

1.20) How do objects pass messages in Java? Select the one correct answer.
(a) They pass messages by modifying each other's fields.
(b) They pass messages by modifying the static variables of each other's classes.
(c) They pass messages by calling each other's instance methods.
(d) They pass messages by calling static methods of each other's classes.

1.21) Given the following code, which statements are true?

```
class A {
int value1;
}
class B extends A {
int value2;
}
```

Select the two correct answers.
(a) Class A extends class B.
(b) Class B is the superclass of class A.
(c) Class A inherits from class B.
(d) Class B is a subclass of class A.
(e) Objects of class A have a field named value2.
(f) Objects of class B have a field named value1.

1.22) What command in the Java 2 SDK should be used to compile the following code contained in a file called SmallProg.java?

```
public class SmallProg {
public static void main(String[] args) { System.out.println("Good luck!");
}
}
```

Select the one correct answer.
(a) java SmallProg
(b) javac SmallProg
(c) java SmallProg.java
(d) javac SmallProg.java
(e) java SmallProg main

1.23) What command in the Java 2 SDK should be used to execute them ain() method of a class named SmallProg? Select the one correct answer.
(a) a. java SmallProg
(b) b. javac SmallProg
(c) c. java SmallProg.java
(d) d. java SmallProg.class
(e) e. java SmallProg.main()

1.24) Which of the following is not a legal identifier? Select the one correct answer.
(a) a. a2z
(b) b. ödipus
(c) c. 52pickup
(d) d. _class
(e) e. ca$h
(f) f. total#

1.25) Which statement is true? Select the one correct answer.

(a) a. new and delete are keywords in the Java language.
(b) b. try, catch, and thrown are keywords in the Java language.
(c) c. static, unsigned, and long are keywords in the Java language.
(d) d. exit, class, and while are keywords in the Java language.
(e) e. return, goto, and default are keywords in the Java language.
(f) f. for, while, and next are keywords in the Java language.

1.26) Is this a complete and legal comment?
```
/* // */
```
Select the one correct answer.
(a) a. No, the block comment (/* ... */) is not ended since the single-line comment (// ...) comments out the closing part.
(b) b. It is a completely valid comment. The/ / part is ignored by the compiler.
(c) c. This combination of comments is illegal and the compiler will reject it.

1.27) Which of the following do not denote a primitive data value in Java? Select the two correct answers.
(a) a. "t"
(b) b. 'k'
(c) c. 50.5F
(d) d. "hello"
(e) e. false

1.28) Which of the following primitive data types are not integer types? Select the three correct answers.
(a) a. Type boolean
(b) b. Type byte
(c) c. Type float
(d) d. Type short
(e) e. Type double

1.29) Which integral type in Java has the exact range from -2147483648 (-231) to 2147483647 (231-1), inclusive? Select the one correct answer.
(a) a. Type byte
(b) b. Type short
(c) c. Type int
(d) d. Type long
(e) e. Type char

1.30) Which of the following lines are valid declarations? Select the three correct answers.
(a) a. char a = '\u0061';
(b) b. char 'a' = 'a';
(c) c. char \u0061 = 'a';
(d) d. ch\u0061r a = 'a';
(e) e. ch'a'r a = 'a';

1.31) Given the following code within a method, which statement is true?
```
int a, b;
b = 5;
```
Select the one correct answer.
(a) a. Local variable a is not declared.
(b) b. Local variable b is not declared.
(c) c. Local variable a is declared but not initialized.
(d) d. Local variable b is declared but not initialized.
(e) e. Local variable b is initialized but not declared.

1.32) In which of these variable declarations will the variable remain uninitialized unless explicitly initialized?
Select the one correct answer.
(a) a. Declaration of an instance variable of type int.
(b) b. Declaration of a static variable of type float.
(c) c. Declaration of a local variable of typef loat.
(d) d. Declaration of a static variable of type Object.

(e) e. Declaration of an instance variable of type int[].

1.33) What will be the result of attempting to compile this class?

```
import java.util.*;
package com.acme.toolkit;
public class AClass {
public Other anInstance;
}
class Other {
int value;
}
```

Select the one correct answer.

(a) a. The class will fail to compile, since the class Other has not yet been declared when referenced in class AClass.
(b) b. The class will fail to compile, since import statements must never be at the very top of a file.
(c) c. The class will fail to compile, since the package declaration can never occur after an import statement.
(d) d. The class will fail to compile, since the class Other must be defined in a file calledO ther.java.
(e) e. The class will fail to compile, since the class Other must be declared public.
(f) f. The class will compile without errors.

1.34) Is an empty file a valid source file? Answer true or false.

1.35) Which of these are valid declarations of the main() method in order to start the execution of a Java application? Select the two correct answers.

(a) a. static void main(String[] args) { /* ... */ }
(b) b. public static int main(String[] args) { /* ... */ }
(c) c. public static void main(String args) { /* ... */ }
(d) d. final public static void main(String[] arguments) { /* ... */ }
(e) e. public int main(Strings[] args, int argc) { /* ... */ }
(f) f. static public void main(String args[]) { /* ... */ }

1.36) Which of the following are reserved keywords? Select the three correct answers.

(a) a. public
(b) b. static
(c) c. void
(d) d. main
(e) e. String
(f) f. args

Language Fundamentals Answers

1.1) (a), (b), (c), (d)
All are valid declarations. The \u*xxxx* notation can be used anywhere in the source to represent Unicode characters, and also casted integer to a char primitive type.

1.2) (a)
Although nonsensical, an empty file ia a valid source file. A source file can contain an optional package declaration, any number of import statements and any number of class and interface definitions.

1.3) (d)
A valid declaration of the main() method must be public and static, have void as return type and take a single array of String objects as arguments. The order of the static and public keywords is irrelevant. Also, declaring the method final does not affect the method's potential to be used as a main() method.

1.4) (a), (d)
You cannot use five digits after \u."char c = 65" is valid because it will take it as ascii value. '\101' is representing a octal value which is equivalent to 65 which is valid but '\1001' is not valid as its decimal value is 513 and you can give only those values which represent 0 to 255 in decimal.

1.5) (d)
This will compile and print "Hello"
The entry point for a standalone java program is the main method of the class that is being run. The java runtime system will look for that method in class Test and find that it does have such a method. It does not matter whether it is defined in the class itself or is inherited from a parent class.

1.6) (a)
This will compile and print "Hello".
The entry point for a standalone java program is the main method of the class that is being run. The java runtime system will look for that method in class Test and find that it does have such a method. It does not matter whether it is defined in the class itself or is inherited from a parent class.

1.7) (b)
The result of 1/4 will be zero because integer divison is carried out on the operands.
If you need to obtain a fractional value you need to use either a float or double literal as in 1F / 4F.

1.8) (a), (c)
(a) is correct as it is a valid hexadecimal number.
(b) is wrong because you can only assign the values true and false to them
(d) is wrong because 128 is beyond the range of a byte.
(e) is wrong because "A" is not a char it is a String.

1.9) (b)
You are passing a reference to an array as the argument to the method. The method may not modify the passed object reference but it can modify the object itself.

1.10) (c)
This will run, print a message and terminate gracefully. The runtime system needs to load the class before it can look for the main method. So the static initializer will run first and print "10". Immediately after that System.exit(0) will be called terminating the program before an error can be thrown.

1.11) (b)
You cannot assign a reference to an array of primitives to another unless they contain the same primitive types.

1.12) (b)
The statement is false. The range of an array is from - 128 to 127

1.13) (a), (b), (d), (e)

(a) is correct because \u0038 is unicode for nbr 8.
(c) is wrong because 1.2 is a double literal.
(d) is a little sneaky perhaps. The unicode escape character is incorrect

1.14) (d)
Division by zero on integer literals will throw a runtime error.

1.15) (d)
A method is an operation defining the behavior for a particular abstraction. Java implements abstractions, using classes that have properties and behavior. Behavior is dictated by the operations of the abstraction.

1.16) (b)
An object is an instance of a class. Objects are created from class definitions that implement abstractions. The objects that are created are concrete realizations of those abstractions.

1.17) (b)
The code marked with (2) is a constructor. A constructor in Java is declared like a method, except that the name is identical to the class name and it does not specify a return value.

1.18) (b) and (f)
Two objects and three reference variables are created by the code. Objects are typically created by using the new operator. Declaration of a reference variable creates a variable regardless of whether a reference value is assigned to it or not.

1.19) (d)
An instance member is a field or an instance method. These members belong to an instance of the class rather than the class as a whole. Members which are not explicitly declared static in a class definition are instance members.

1.20) (c)
An object can pass a message to another object by calling an instance method of the other object.

1.21) (d) and (f)
Given the declaration class B extends A {...} we can conclude that class B extends class A, class A is the superclass of class B, class B is a subclass of class A, and class B inherits from class A, which means that objects of class B will inherit the field value1 from class A.

1.22) (d)
The compiler supplied with the Java 2 SDK is named javac. It requires the names of the source files that should be compiled.

1.23) (a)
Java programs are executed by the Java Virtual Machine (JVM). In the Java 2 SDK, the command java is used to start the execution by the JVM. The java command requires the name of a class that has a validm ain() method. The JVM starts the program execution by calling the main() method of the given class. The exact name of the class should be specified, and not the name of the class file, that is, the ".class" extension in the class file name should not be specified.

1.24) (c)
52pickup is not a legal identifier. The first character of an identifier cannot be a digit.

1.25) (e)
In Java, the identifiers delete, thrown, exit, unsigned, and next are not keywords. Java has a goto keyword, but it is
reserved and not currently used.

1.26) (b)
It is a completely valid comment. Comments do not nest. Everything from the start marker of a comment block (/*) until
the first occurrence of the end marker of the comment block (*/) is ignored by the compiler.

1.27) (a) and (d)
String is a class, and "hello" and "t" denote String objects. Java only has the following primitive data types: boolean, byte,
short, char, int, long, float, and double.

1.28) (a), (c), and (e)
Type (a) is a boolean data type, while types (c) and (e) are floating-point data types.

1.29) (c)
The bit representation of int is 32-bits wide and can hold values in the range 231 through 231 -1.

1.30) (a), (c), and (d)
The \uxxxx notation can be used anywhere in the source to represent unicode characters.

1.31) (c)
Local variable a is declared but not initialized. The first line of code declares the local variablesa and b. The second line of code initializes the local variable b. Local variable a remains uninitialized.

1.32) (c)
The local variable of type float will remain uninitialized. Fields receive a default value unless explicitly initialized. Local variables remain uninitialized unless explicitly initialized. The type of the variable does not affect whether a variable is initialized or not.

1.33) (c)
The class will fail to compile since the package declaration can never occur after an import statement. The package and import statements, if present, must always precede any class definitions. If a file contains bothim port statements and a package statement, then the package statement must occur before the import statements.

1.34) true
Although nonsensical, an empty file is a valid source file. A source file can contain an optional package declaration, any number of import statements, and any number of class and interface definitions.

1.35) (d) and (f)
The main() method must be declared public, static, and void and takes a single array of String objects as argument. The order of the static and public keywords is irrelevant. Also, declaring the method final is irrelevant in this respect.

1.36) (a), (b), and (c)
Neither main, string, nor args are reserved keywords, but they are legal identifiers. In the declaration public static void main(String[] args), the identifier main denotes the method that is the main entry point of a program. In all other contexts, the identifier main has no predefined meaning.

Operator & Assignments Mock Questions

2.1) Which statements about the output of the following program are true?

```
public class Logic {
    public static void main(String args[]) {
        int i = 0;
        int j = 0;

        boolean t = true;
        boolean r;

        r = (t && 0<(i+=1));
        r = (t && 0<(i+=2));
        r = (t && 0<(j+=1));
        r = (t || 0<(j+=2));

        System.out.println( i + "    " + j );
    }
}
```

(a) The first digit printed is 1.
(b) The first digit printed is 2.
(c) The first digit printed is 3.
(d) The second digit printed is 1.
(e) The second digit printed is 2.
(f) The second digit printed is 3.

2.2) Which statements about the output of the following program are true?

```
public static void main(String args[])
{
    int i =0;
    i = i++;
    System.out.println(i);
}
```

(a) 0 is printed.
(b) 1 is printed.

2.3) Which statements about the output of the following program are true?

```
public class EqualTest {
    public static void main(String args[]) {
        String s1 = "YES";
        String s2 = "YES";
        if ( s1 == s2 ) System.out.println("equal");
        String s3 = new String("YES");
        String s4 = new String("YES");
        if ( s3 == s4 ) System.out.println("s3 eq s4");
    }
}
```

(a) "equal" is printed, "s3 eq s4" is printed.
(b) "equal" is printed only.
(c) "s3 eq s4" is printed only.
(d) Nothing is printed.

2.4) What happens when you try to compile and run the following code?

```
public class EqualsTest {
    public static void main(String args[]) {
        char A = '\u0005';
        if ( A == 0x0005L )
            System.out.println("Equal");
        else
            System.out.println("Not Equal");
```

```
        }
    }
```

(a) The compiler reports "Invalid character in input" in line 3
(b) The program compiles and prints "Not equal".
(c) The program compiles and prints "Equal".

2.5) What will happen when you attempt to compile and run the following code?

```
public class As{
    int i = 10;
    int j;
    char z= 1;
    boolean b;

    public static void main(String argv[]){
        As a = new As();
        a.amethod();
    }

    public void amethod(){
        System.out.println(j);
        System.out.println(b);
    }
}
```

(a) Compilation succeeds and at run time an output of 0 and false.
(b) Compilation succeeds and at run time an output of 0 and true.
(c) Compile time error b is not initialised.
(d) Compile time error z must be assigned a char value.

2.6) Given the following code what will be the output?

```
class ValHold {
    public int i = 10;
}

public class ObParm {
    public static void main(String argv[]) {
        ObParm o = new ObParm();
        o.amethod();
    }

    public void amethod(){
        int i = 99;
        ValHold v = new ValHold();
        v.i=30;
        another(v,i);
        System.out.println(v.i);
    } //End of amethod

    public void another(ValHold v, int i){
        i = 0;
        v.i = 20;
        ValHold vh = new ValHold();
        v =  vh;
        System.out.println(v.i+ " "+i);
    }//End of another
}
```

(a) 10,0,30
(b) 20,0,30
(c) 20,99,30
(d) 10,0,20

2.7) Here are three proposed alternatives to be used in a method to return false if the object reference x has the null value. Which statement will work?

(a) if (x == null) return false;
(b) if (x.equals(null)) return false;
(c) if (x instanceof null) return false;

2.8) What will be the result of compiling and running the given program?
Select one correct answer.

```
1 class Q1
2 {
3     public static  void main(String arg[])
4     {
5        int a[]={2,2};
6        int b=1;
7        a[b]=b=0;
8        System.out.println(a[0]);
9        System.out.println(a[1]);
10    }
11 }
```

(a) Compile time error at the line no. 5.
(b) Run time error at the line no. 5.
(c) Program compiles correctly and print 2,0 when executed.
(d) Program compiles correctly and print 0,2 when executed.

2.9) What will be the result of compiling and running the given program?
Select one correct answer.

```
1 public class Q4
2 {
3    public static void main(String[] args)
4    {
5       boolean t1 = true, t2 = false, t3 = t1;
6       boolean t = false;
7       t &&= (t1 || ( t2 && t3));
8       System.out.println(t);
9    }
10 }
```

(a) Program compiles correctly and print true when executed.
(b) Program compiles correctly and print false when executed.
(c) Compile time error.
(d) Run time error.

2.10) What will be the result of compiling and running the given program?
Select one correct answer.

```
1 class AA{}
2 class BB extends AA{}
3 class Q6
4 {
5    public static void main(String arg[])
6    {
7       AA a=null;
8       BB b=(BB)a;
9       System.out.println(b);
10      System.out.println(b instanceof BB);
11      System.out.println(b instanceof AA);
12   }
13 }
```

(a) Program compiles correctly and print null,true,false.
(b) Program compiles correctly and print null,true,true.
(c) Program compiles correctly and print null,false,false.
(d) Program compiles correctly and print null,false,true.
(e) Compile time error at line no.8 as null value cannot be casted.
(f) Run time error at line no. 8 as null value cannot be casted.

2.11) Which one of the following are valid character contants?
Select any two.
(a) char c = '\u000d' ;
(b) char c = '\u000a';
(c) char c = '\u0000';
(d) char c = '\uface' ;

2.12) Given char c = 'A';
What is the simplest way to convert the character value in c into an int? Select the one correct answer.
(a) int i = c;
(b) int i = (int) c;
(c) int i = Character.getNumericValue(c);

2.13) What will be the result of attempting to compile and run the following class?

```
public class Assignment {
public static void main(String[] args) {
int a, b, c;
b = 10;
a = b = c = 20;
System.out.println(a);
}
}
```

Select the one correct answer.
(a) The code will fail to compile, since the compiler will recognize that the variable c in the assignment statement a = b = c = 20; has not been initialized.
(b) The code will fail to compile because the assignment statementa = b = c = 20; is illegal.
(c) The code will compile correctly and will display1 0 when run.
(d) The code will compile correctly and will display2 0 when run.

2.14) What will be the result of attempting to compile and run the following program?

```
public class MyClass {
public static void main(String[] args) {
String a, b, c;
c = new String("mouse");
a = new String("cat");
b = a;
a = new String("dog");
c = b;
System.out.println(c);
}
}
```

Select the one correct answer.
(a) The program will fail to compile.
(b) The program will print mouse when run.
(c) The program will print cat when run.
(d) The program will print dog when run.
(e) The program will randomly print either cat or dog when run.

2.15) Which of the following expressions will be evaluated using floating-point arithmetic? Select the three correct answers.
(a) 2.0 * 3.0
(b) 2 * 3
(c) 2/3 + 5/7
(d) 2.4 + 1.6
(e) 0x10 * 1L * 300.0

2.16) What is the value of the expression (1 / 2 + 3 / 2 + 0.1)? Select the one correct answer.
(a) 1
(b) 1.1
(c) 1.6
(d) 2

(e) 2.1

2.17) What will be the result of attempting to compile and run the following program?

```
public class Integers {
public static void main(String[] args) {
System.out.println(0x10 + 10 + 010);
}
}
```

Select the one correct answer.

(a) The program will not compile. The compiler will complain about the expression 0x10 + 10 + 010.
(b) When run, the program will print 28.
(c) When run, the program will print 30.
(d) When run, the program will print 34.
(e) When run, the program will print 36.
(f) When run, the program will print 101010.

2.18) Which of the following expressions are valid? Select the three correct answers.

(a) (- 1 -)
(b) (+ + 1)
(c) (+-+-+-1)
(d) (--1)
(e) (1 * * 1)
(f) (- -1)

2.19) What is the value of evaluating the following expression (- -1-3 * 10 / 5-1)? Select the one correct answer.

(a) -8
(b) -6
(c) 7
(d) 8
(e) 10
(f) None of the above.

2.20) Which of these assignments are valid? Select the four correct answers.

(a) short s = 12;
(b) long l = 012;
(c) int other = (int) true;
(d) float f = -123;
(e) double d = 0x12345678;

2.21) Which statements are true? Select the three correct answers.

(a) The result of the expression (1 + 2 + "3") would be the string "33".
(b) The result of the expression ("1" + 2 + 3) would be the string "15".
(c) The result of the expression (4 + 1.0f) would be the float value 5.0f.
(d) The result of the expression (10/9) would be the int value 1.
(e) The result of the expression ('a' + 1) would be the char value 'b'.

2.22) What happens when you try to compile and run the following program?

```
public class Prog1 {
public static void main(String[] args) {
int k = 1;
int i = ++k + k++ + + k;
System.out.println(i);
}
}
```

Select the one correct answer.

(a) The program will not compile. The compiler will complain about the expression ++k + k++ + + k.
(b) The program will compile and will print the value3 when run.
(c) The program will compile and will print the value4 when run.
(d) The program will compile and will print the value7 when run.

(e) The program will compile and will print the value8 when run.

2.23) Which is the first incorrect line that will cause a compile time error in the following program?

```
public class MyClass {
public static void main(String[] args) {
char c;
int i;
c = 'a'; // (1)
i = c; // (2)
i++; // (3)
c = i; // (4)
c++; // (5)
}
}
```

Select the one correct answer.

(a) The line labeled (1)
(b) The line labeled (2)
(c) The line labeled (3)
(d) The line labeled (4)
(e) The line labeled (5)
(f) None of the lines are incorrect. The program will compile just fine.

2.24) What happens when you try to compile and run the following program?

```
public class Cast {
public static void main(String[] args) {
byte b = 128;
int i = b;
System.out.println(i);
}
}
```

Select the one correct answer.

(a) The compiler will refuse to compile it, since you cannot assign ab yte to an int without a cast.
(b) The program will compile and will print1 28 when run.
(c) The compiler will refuse to compile it, since1 28 is outside the legal range of values for ab yte.
(d) The program will compile, but will throw aC lassCastException when run.
(e) The program will compile and will print2 55 when run.

2.25) What will the following program print when run?

```
public class EvaluationOrder {
public static void main(String[] args) {
int[] array = { 4, 8, 16 };
int i=1;
array[++i] = --i;
System.out.println(array[0] + array[1] + array[2]);
}
}
```

Select the one correct answer.

(a) 13
(b) 14
(c) 20
(d) 21
(e) 24

2.26) Which of the following expressions evaluates to true? Select the two correct answers.

(a) (false | true)
(b) (null != null)
(c) (4 <= 4)
(d) (!true)
(e) (true & false)

2.27) Which statements are true? Select the two correct answers.

(a) The remainder operator % can only be used with integral operands.
(b) Identifiers in Java are case insensitive.
(c) The arithmetic operators *, /, and % have the same level of precedence.
(d) A short value ranges from -128 to +127 inclusive.
(e) (+15) is a legal expression.

2.28) Which statements are true about the lines of output printed by the following program?

```
public class BoolOp {
static void op(boolean a, boolean b) {
boolean c = a != b;
boolean d = a ^ b;
boolean e = c == d;
System.out.println(e);
}
public static void main(String[] args) {
op(false, false);
op(true, false);
op(false, true);
op(true, true);
}
}
```

Select the three correct answers.
(a) All lines printed are the same.
(b) At least one line contains false.
(c) At least one line contains true.
(d) The first line contains false.
(e) The last line contains true.

2.29) What happens during execution of the following program?

```
public class OperandOrder {
public static void main(String[] args) {
int i = 0;
int[] a = {3,6};
a[i] = i = 9;
System.out.println(i + " " + a[0] + " " + a[1]);
}
}
```

Select the one correct answer.
(a) Throws an exception of type ArrayIndexOutOfBoundsException
(b) Prints "9 9 6"
(c) Prints "9 0 6"
(d) Prints "9 3 6"
(e) Prints "9 3 9"

2.30) Which statements are true about the output of the following program?

```
public class Logic {
public static void main(String[] args) {
int i = 0;
int j = 0;
boolean t = true;
boolean r;
r = (t & 0<(i+=1));
r = (t && 0<(i+=2));
r = (t | 0<(j+=1));
r = (t || 0<(j+=2));
System.out.println(i + " " + j);
}
}
```

Select the two correct answers.
(a) The first digit printed is 1.
(b) The first digit printed is 2.
(c) The first digit printed is 3.

(d) The second digit printed is 1.
(e) The second digit printed is 2.
(f) The second digit printed is 3.

2.31) What would be printed during execution of the following program?

```
public class MyClass {
public static void main(String[] args) {
test(1<<32, "1<<32");
test(1<<31, "1<<31");
test(1<<30, "1<<30");
test(1, "1" );
test(0, "0" );
test(-1, "-1" );
}
public static void test(int i, String exp) {
if ((i >> 1) != (i >>> 1)) System.out.println(exp);
}
}
```

Select the two correct answers.
(a) "1<<32"
(b) "1<<31"
(c) "1<<30"
(d) "1"
(e) "0"
(f) "-1"

2.32) Which of the following are not operators in Java? Select the two correct answers.
(a) %
(b) <<<
(c) &
(d) %=
(e) >>>
(f) <=
(g) &&=

2.33) Given a variable x of type int (which may contain a negative value), which are correct ways of doubling the value oxf , barring any wrapping of out-of-range intermediate values? Select the four correct answers.
(a) x << 1;
(b) x = x * 2;
(c) x *= 2;
(d) x += x;
(e) x <<= 1;

2.34) Which of the following operators can be used both as an integer bitwise operator and a boolean logical operator? Select the three correct answers.
(a) ^
(b) !
(c) &
(d) |
(e) ~

2.35) Given these declarations, which of the following expressions are valid?

```
byte b = 1;
char c = 1;
short s = 1;
int i = 1;
```

Select the three correct answers.
(a) s = b * 2;
(b) i = b << s;
(c) b <<= s;

(d) c = c + b;
(e) s += i;

2.36) What will be printed when the following program is run?

```
public class ParameterPass {
public static void main(String[] args) {
int i = 0;
addTwo(i++);
System.out.println(i);
}
static void addTwo(int i) {
i += 2;
}
}
```

Select the one correct answer.

(a) a. 0
(b) b. 1
(c) c. 2
(d) d. 3

2.37) What will be the result of attempting to compile and run the following class?

```
public class Passing {
public static void main(String[] args) {
int a = 0; int b = 0;
int[] bArr = new int[1]; bArr[0] = b;
inc1(a); inc2(bArr);
System.out.println("a=" + a + " b=" + b + " bArr[0]=" + bArr[0]);
}
public static void inc1(int x) { x++; }
public static void inc2(int[] x) { x[0]++; }
}
```

Select the one correct answer.

(a) The code will fail to compile, since x[0]++; is not a legal statement.
(b) This document was created by an unregistered ChmMagic, please go to http://www.bisenter.com to regis.ter it. Thanks
(c) The code will compile and will print "a=1 b=1 bArr[0]=1" when run.
(d) The code will compile and will print" a=0 b=1 bArr[0]=1" when run.
(e) The code will compile and will print" a=0 b=0 bArr[0]=1" when run.
(f) The code will compile and will print" a=0 b=0 bArr[0]=0" when run.

2.38) Given the class

```
// Filename: Args.java
public class Args {
public static void main(String[] args) {
System.out.println(args[0] + " " + args[args.length-1]);
}
}
```

what would be the result of executing the following on the command line? Select the one correct answer.

(a) The program will throw ArrayIndexOutOfBoundsException.
(b) The program will print "java handicap".
(c) The program will print "Args handicap".
(d) The program will print "In handicap".
(e) The program will print "Args a".
(f) The program will print "In a".

2.39) Which statements would cause a compilation error if inserted in the location indicated in the following program?

```
public class ParameterUse {
static void main(String[] args) {
int a = 0;
final int b = 1;
```

```
int[] c = { 2 };
final int[] d = { 3 };
useArgs(a, b, c, d);
}
static void useArgs(final int a, int b, final int[] c, int[] d) {
// INSERT STATEMENT HERE.
}
}
```

Select the two correct answers.

(a) a++;
(b) b++;
(c) b = a;
(d) c[0]++;
(e) d[0]++;
(f) c = d;

Operators & Assignments Answers

2.1) (c), (d)
Unlike & and | operators, the && and || operators short circuit the evaluation of their operands if the result of the operation can be determined just based on the value of the first operand. The second operand of the || operation in the program is never evaluated. Variable i ends up with the value 3 which is the first digit printed, and j ends up with a value of 1 which is the second digit printed.

2.2) (a)
To explain u in a more simple way...

int k = 10;
int j;

Post increment :
j = k++;

In this statement the value of k is 10, & there is a ++ operator also attached with it, so it will be assigned to j, so now j gets 10, but k becomes as 11 because the ++ expression comes after the variable, or if u see k as k++.

Pre-increment :
j = ++k;
In this case, k is having 10 as value, then the pre-increment operation is being performed, so k becomes 11 in this case, & then it is being assigned to j, so now j becomes 11 & k also becomes 11.

The thing which u have must have noticed in the post increment & pre-increment is that the value of k is increased by 1, where as j in the first case becomes 10 & in second case j becomes 11.

2.3) (b)
The compiler creates one String object for both s1 and s2, thus "equal" appears. But using the new String operator two distinct objects are created so "s3 eq s4" does not appear.

2.4) (c)
The compiler promotes variable A to a long before the comparison. The compiler does not report "Invalid character in input" in line 3 because this is correct form for initializing a char primitive. Therefore, answer (a) is incorrect. Because (c) is correct ⇒ (b) cannot possibly be correct.

2.5) (a)
The default value for a boolean declared at class level is false, and integer is 0, and 1 is within the range of the char and don't need casting.

2.6) (d)
In the call another(v,i);
A **COPY** of the reference to v is passed and thus any changes will be intact after this call.

2.7) (a)
The **ONLY** correct way to check a reference for the null value. Answer (b) is incorrect because if x is null, there will not be an object whose equals method can be called. This statement would cause a NullPointerException at runtime when x is null. Answer (c) is incorrect because only a reference type, such as a class, or array, can be the right operand of the instanceof operator.

2.8) (c)
First of all value of b which is 1 is assigned to array index then rest of the code will be executed.

2.9) (c)

Compile time error: There is no operator like &&=

2.10) (c)
As null value can be casted but do remember it is of no use to cast null value that is why it is written in Khalid Mughal's book that we can not cast null value.Secondly,instanceof operator always return false for null value.

2.11) (c), (d)
'\u000d' and '\u000a' are not valid as these value do line break. Note:'\u000a' is not valid even when used behind single line comments (i.e. //.....),but offcourse for multiline comment it is valid. '\uface' is valid because we can use any character from range 0 - 9 or A - F or a - f.

2.12) (a)
A value of type char can be assigned to a variable of type int. An implicit widening conversion will convert the value to an int.

2.13) (d)
An assignment statement is an expression statement. The value of the expression statement is the value of the expression on the right hand side. Since the assignment operator is right associative, the statement a = b = c = 20 is evaluated as follows: (a = (b = (c = 20))). This results in the value 20 being assigned to variablec , then the same value being assigned to variable b and finally to variable a. The program will compile correctly and display 20 when run.

2.14) (c)
Strings are objects. The variables a, b, and c are references that can denote such objects. Assigning to a reference only changes the reference value. It does not create a copy of the source object or change the object denoted by the old reference value in the destination reference. In other words, assignment to references only affects which object the destination reference denotes. The reference value of the "cat" object is first assigned to variable a, then to variable b, and later to variable c. The program prints the string denoted by the variable c, which is "cat".

2.15) (a), (d), and (e)
A binary expression with any floating-point operand will be evaluated using floating-point arithmetic. Expressions such as 2/3, where both operands are integers, will use integer arithmetic and evaluate to an integer value.

2.16) (b)
The / operator has higher precedence than the + operator. This means that the expression is evaluated as((1/2) + (3/2) + 0.1). The associativity of the binary operators is from left to right, giving (((1/2) + (3/2)) + 0.1). Integer division results in ((0 + 1) + 0.1) which evaluates to 1.1.

2.17) (d)
0x10 is a hexadecimal literal equivalent to the decimal value 16. 10 is a decimal literal. 010 is an octal literal equivalent to the decimal value 8. The println() method will print the sum of these values, which is 34, in decimal form.

2.18) (b), (c), and (f)
The unary + and - operators with right-to-left associativity are used in the valid expressions (b), (c), and (f). Expression (a) tries to use a nonexistent unary - operator with left-to-right associativity, expression (d) tries to use a decrement operator (--) on an expression that does not resolve to a variable, and expression (e) tries to use a nonexistent unary* operator.

2.19) (b)
The expression evaluates to –6. The whole expression is evaluated as (((-(-1)) - ((3 * 10) / 5)) - 1) according to the precedence and associativity rules.

2.20) (a), (b), (d), and (e)
In (a) the conditions for implicit narrowing conversion are fulfilled: the source is a constant expression of type int, the destination type is of type short, the value of the source (12) is in the range of the destination type. The assignments in (b), (d), and (e) are valid, since the source type is narrower than the target type

and an implicit widening conversion will be applied. The expression (c) is not valid. Values of type boolean cannot be converted to other types.

2.21) (a), (c), and (d)
The left associativity of the + operator makes the evaluation of (1 + 2 + "3") proceed as follows: (1 + 2) + "3" 3 + "3" "33". Evaluation of the expression ("1" + 2 + 3), however, will proceed as follows: ("1" + 2) + 3 "12" + 3 "123". (4 + 1.0f) evaluates as 4.0f + 1.0f 5.0f and (10/9) performs integer division, resulting in the value 1. The operand 'a' in the expression ('a' + 1) will be promoted to int, and the resulting value will be of type int.

2.22) (d)
The expression ++k + k++ + + k is evaluated as ((++k) + (k++)) + (+k) ((2) + (2) + (3)), resulting in the value 7.

2.23) (d)
The types char and int are both integral. A char value can be assigned to an int variable since the int type is wider than the char type and an implicit widening conversion will be done. An int type cannot be assigned to a char variable because the char type is narrower than the int type. The compiler will report an error about a possible loss of precision in the line labeled (4).

2.24) (c)
Variables of type byte can store values in the range –128 to 127. The expression on the right-hand side of the first assignment is the int literal 128. Had this literal been in the range of theb yte type, an implicit narrowing conversion would have to be applied during assignment to convert it to a byte value. Since 128 is outside the valid range of type byte, the compiler will not compile the code.

2.25) (a)
First, the expression ++i is evaluated, resulting in the value 2, Now the variable i also has the value 2. The target of the assignment is now determined to be the element array[2]. Evaluation of the right-hand expression, --i, results in the value 1. The variable i now has the value 1. The value of the right-hand expression 1 is then assigned to the array element array[2], resulting in the array contents to become {4, 8, 1}. The program sums these values and prints 13.

2.26) (a) and (c)
The expression (4 <= 4) is true. The null literal can be compared, so (null != null) yields false.

2.27) (c) and (e)
The remainder operator is not limited to integral values, but can also be applied to floating-point operands. Identifiers in Java are case sensitive. Operators *, /, and % have the same level of precedence. Type short has the range -32768 to +32767 inclusive. (+15) is a legal expression using the unary + operator.

2.28) (a), (c), and (e)
The != and ^ operators, when used on boolean operands, will return true if and only if one operand is true, and false otherwise. This means that d and e in the program will always be assigned the same value, given any combination of truth values in a and b. The program will, therefore, print true four times.

2.29) (b)
The element referenced by a[i] is determined based on the current value of i, which is zero, that is, the element a[0]. The expression i = 9 will evaluate to the value 9, which will be assigned to the variable i. The value 9 is also assigned
to the array element a[0]. After the execution of the statement, the variable i will contain the value 9, and the array a will contain the values 9 and 6. The program will print 9 9 6 when run.

2.30) (c) and (d)
Unlike the & and | operators, the && and || operators short-circuit the evaluation of their operands if the result of the operation can be determined from the value of the first operand. The second operand of the || operator in the program is never evaluated because of short-circuiting. All the operands of the other operators are evaluated. Variable i ends up with a value of 3, which is the first digit printed, and j ends up with a value of 1, which is the second digit printed.

2.31) (b) and (f)
The method test() will print out its second argument if the results of performing a signed and an unsigned 1-bit right shift on its first argument differ. The only difference between these operations is that when performing a signed shift, the leftmost bit will retain its state, rather than being assigned the bit value 0. The operational difference will, therefore, only be apparent when applied on values where the value of the leftmost bit is 1. Of the values being passed to the method test(), only the result of the expression 1<<31 (i.e., 1000 ... 0000) and the value -1 (1111 ... 1111) have the left-most bit set.

2.32) (b) and (g)
Java has the operators >> and >>> to perform signed and unsigned right shifts. For left shifts there is no difference between shifting signed and unsigned values. Java, therefore, only has one left-shift operator, which is <<. <<< is not an operator in Java. Java has the boolean AND compound assignment operator &=, but &&= is not an operator in Java.

2.33) (b), (c), (d), and (e)
All the expressions will return the same result. All expressions will accommodate negative values, and x can be any value of type int. However, expression (a) will not assign the result back to the variable x.

2.34) (a), (c), and (d)
The logical complement operator (!) cannot be used as an integer bitwise operator, and the bitwise complement operator (~) cannot be used as a boolean logical operator.

2.35) (b), (c), and (e)
All the values of the expressions on the right-hand side of the assignments are implicitly promoted to type int. For expression (b) this works, since the target type is also int. The compound assignment operators in expressions (c) and (e) ensure that an implicit narrowing conversion makes the result fit back in the target variable. Expressions (a) and (d) are simply invalid, since the type of expression on the right-hand side of the assignment operator is not compatible with the type of the target variable on the left-hand side.

2.36) (b)
Evaluation of the actual parameter i++ yields 0, and increments i to 1 in the process. The value 0 is copied into the formal parameter i of the method addTwo() during method invocation. However, the formal parameter is local to the method, and changing its value does not affect the value in the actual parameter. The value of variable i in the main() method remains 1.

2.37) (d)
The variables a and b are local variables that contain primitive values. When these variables are passed as parameters to another method, the method receives copies of the primitive values in the variables. The original variables are unaffected by operations performed on the copies of the primitive values within the called method. The variable bArr contains a reference value that denotes an array object containing primitive values. When the variable is passed as a parameter to another method, the method receives a copy of the reference value. Using this reference value, the method can manipulate the object that the reference value denotes. This allows the elements in the array object referenced by bArr to be accessed and modified in the method inc2().

2.38) (d)
The length of the array passed to the main() method corresponds exactly to the number of command-line arguments given to the program. Unlike some other programming languages, the element at index 0 does not contain the name of the program. The first argument given is retrieved using args[0], and the last argument given is retrieved using args[args.length-1].

2.39) (a) and (f)
Values can only be assigned once to final variables. A final formal parameter is assigned the value of the actual parameter at method invocation. Within the method body, it is illegal to reassign or modify the value of a final parameter. This causes a++ and c = d to fail. Whether the actual parameter is final does not constrain the client that invoked the method, since the actual parameter values are copied to the formal parameters.

Declarations & Access Control Mock Questions

3.1) Which of these array declarations and instantiations are not legal?
Select all valid answers:
(a) int []a[] = new int[4][4];
(b) int a[][] = new int[4][4];
(c) int a[][] = new int[][4];
(d) int []a[] = new int[4][];
(e) int[][] a = new int[4][4];
(f) int [ ] [ ] a = new int[4][4];

3.2) Which of these array declarations and initialization are not legal?
Select all valid answers:
(a) int []i[] = {{1,2}, {1}, {}, {1,2,3}};
(b) int i[]= new int[2]{1, 2};
(c) int i[][] = new int[][]{{1,2,3}, {4,5,6}};
(d) int i[][] = {{1, 2}, new int[2]};
(e) int i[4] = {1, 2, 3, 4};

3.3) Given the following code, which statements can be placed at the indicated position without causing compilation errors?

```
public class ThisUsage {
    int planets;
    static int suns;

    public void gaze() {
        int i;
        // … insert statements here …
    }
}
```

Select all valid answers:
(a) i = this.planets;
(b) i = this.suns;
(c) this = new ThisUsage();
(d) this.i = 4;
(e) this.suns = planets;

3.4) Given the following pairs of method declarations, which of these statements ar true?

```
void fly(int distance) {}
int fly(int time, int speed) {return time*speed}

void fall(int time) {}
int fall(int distance) {return distance}

void glide(int time) {}
void Glide(int time) {}
```

Select all valid answers:
(a) The first pair of methods will compile correctly and overload the method name fly.
(b) The second pair of methods will compile correctly and overload the method name fall.
(c) The third pair of methods will compile correctly and overload the method name glide.
(d) The second pair of methods will not compile correctly.
(e) The third pair of methods will not compile correctly.

3.5) Given a class named Book, which of these would be valid definitions of constructors for the class?
Select all valid answers:
(a) Book(Book b) {}
(b) Book Book() {}
(c) private final Book() {}
(d) void Book() {}

(e) public static void Book(String args[]) {}
(f) abstract Book() {}

3.6) Which of these statements are true?
Select all valid answers:
(a) All classes must define a constructor.
(b) A constructor can be declared private.
(c) A constructor can declare a return value.
(d) A constructor must initialize all the member variables of a class.
(e) A constructor can access the non-static members of a class.

3.7) What will be the result of attempting to compile the following program?

```
Public class MyClass {
    Long var;

    Public void MyClass(long param) { var = param; }    // (1)

    Public static void main( String args[] ) {
        MyClass a, b;
        A = new MyClass();                              // (2)
        B = new MyClass(5);                             // (3)
    }
}
```

Select the one right answer:
(a) A compilation error will be encountered at (1), since constructors should not specify a return value.
(b) A compilation error will be encountered at (2), since the class does not have a default constructor
(c) A compilation error will be encountered at (3), since the class does not have a constructor accepting a single argument of type int.
(d) The program will compile correctly.

3.8) Given the following class, which of these are valid ways of referring to the class from outside of the

```
package net.basemaster?
package net.basemaster;

public class Base {
    // …
}
```

Select all valid answers:
(a) By simply referring to the class as Base.
(b) By simply referring to the class as basemaster.Base
(c) By simply referring to the class as net.basemaster.Base
(d) By importing net.basemaster.* and referring to the class as Base
(e) By importing the package net.* and referring to the class as basemaster.Base

3.9) Which one of the following class definitions is a legal definition of a class that cannot be instantiated?
Select the one right answer:
(a) class Ghost {
    abstract void haunt()
}
(b) abstract class Ghost {
    void haunt();
}
(c) abstract class Ghost {
    void haunt() {};
}
(d) abstract Ghost {
    abstract void haunt();
}
(e) static class Ghost {
    abstract haunt();

```
}
```

3.10) Which of these statements concerning the use of modifiers are true?
Select all valid answers:
(a) If no accessibility modifier (public, protected and private) is given in a member declaration of a class, the member is only accessible for classes in the same package and subclasses of the class.
(b) You cannot specify visibility of local variables. They are always only accessible within the block in which they ardeclared.
(c) Subclasses of a class must reside in the same package as the class they extend.
(d) Local variables can be declared static.
(e) Objects themselves don not have visibility, only references to the object.

3.11) Given the following source code, which one of the lines that are commeneted out may be reinserted without introducing errors?

```
abstract class MyClass {
     abstract void f();
     final void g() {}
     // final void h() {}                    // (1)

    protected static int i;
    private          int j;
}
final class MyOtherClass extends MyClass {
    // MyOtherClass(int n) { m = n; }  // (2)

    public static void main(String args[]) {
        MyClass mc = new MyOtherClass();
    }

    void f() {}
    void h() {}
    // void k() {i++;}                        // (3)
    // void l() {j++;}                        // (4)

    int m;
}
```

Select the one right answer:
(a) (1)
(b) (2)
(c) (3)
(d) (4)

3.12) Which of these statements are true?
Select all valid answers:
(a) A static method can call other non-static methods in the same class by using the this keyword.
(b) A class may contain both static and non-static variables and both static and non-static methods.
(c) Each object of a class has its own instance of each static member variable.
(d) Instance methods may access local variables of static methods.
(e) All methods in a class are implicitly passed a this parameter when called.

3.13) Which of these statements are true?
Select all valid answers:
(a) Transient variables will not be saved during serialization.
(b) Constructors can be declared abstract.
(c) The initial state of an array object constructed with the statement int a[] = new int[10] will depend on whether the variable a[] is a local variable, or a member variable of a class.
(d) Any subclass of a class with an abstract method must implement a method body for that method.
(e) Only static methods can access static members.

3.14) Which of the following are Java modifiers?
(a) public

(b) private
(c) friendly
(d) transient
(e) vagrant

3.15) Why might you define a method as native?
(a) To get to access hardware that Java does not know about.
(b) To define a new data type such as an unsigned integer.
(c) To write optimised code for performance in a language such as C/C++
(d) To overcome the limitation of the private scope of a method

3.16) What will happen when you attempt to compile and run this code?

```
public class Mod{
    public static void main(String argv[]){
    }
    public static native void amethod();
}
```

(a) Error at compilation: native method cannot be static.
(b) Error at compilation native method must return value.
(c) Compilation but error at run time unless you have made code containing native amethod available.
(d) Compilation and execution without error.

3.17) What will happen when you attempt to compile and run this code?

```
private class Base{}
public class Vis{
        transient int  iVal;
        public static void main(String elephant[]){
        }
}
```

(a) Compile time error: Base cannot be private.
(b) Compile time error indicating that an integer cannot be transient.
(c) Compile time error transient not a data type.
(d) Compile time error malformed main method.

3.18) What will happen when you attempt to compile and run the following code?

```
public class Hope{
    public static void main(String argv[]) {
        Hope h = new Hope();
    }

    protected Hope() {
        for(int i =0; i <10; i ++){
            System.out.println(i);
        }
    }
}
```

(a) Compilation error: Constructors cannot be declared protected.
(b) Run time error: Constructors cannot be declared protected.
(c) Compilation and running with output 0 to 10.
(d) Compilation and running with output 0 to 9.

3.19) What will happen when you attempt to compile and run the following code with the command line `"hello there"`?

```
public class Arg{
String[] MyArg;
    public static void main(String argv[]){
        MyArg=argv;
    }

    public void amethod(){
```

```
            System.out.println(argv[1]);
        }
    }
```

(a) Compile time error.
(b) Compilation and output of "hello".
(c) Compilation and output of "there".
(d) None of the above.

3.20) Which of the following method will not give you any error when placed on line no.3? [8]
Select one correct answer.

```
1  interface i2
2  {
3     //Here
4  }
```

(a) static void abc();
(b) abstract void abc();
(c) native void abc();
(d) synchronized void abc();
(e) final void abc();
(f) private void abc();
(g) protected void abc();

3.21) What is the result of attempting to compile and run this

```
interface A{
 void aMethod();
 }
 public class Test implements A{
 void aMethod(){
 System.out.println("hello");
 }
 public static void main(String[] args){
 Test t = new Test();
 t.aMethod();
 }
}
```

(a) The code will not compile.
(b) Runtime exception.
(c) Compiles and runs printing out "hello".

3.22) Is this code legal ?

```
public class Test {
 void aMethod(){
 static int b = 10;
 System.out.println(b);
 }
}
```

(a) Yes
(b) No

3.23) Is this legal

```
public class Test {
  static { int a = 5; }
  public static void main(String[] args){
  System.out.println(a);
  }
 }
```

(a) Yes
(b) No

3.24) Select the true statements.
(a) Transient methods cannot be overridden

(b) A final class may not be subclassed.
(c) A private method can never be overridden to become public
(d) An abstract class may contain final methods
(e) A private method cannot be overridden and made public
(f) Final methods may not be overridden.

3.25) Which of the following are valid for declaring and intialising a char variable?
(a) char c = 'a';
(b) char c = '\'';
(c) char c = '\n';
(d) char c = "a";
(e) char c = \u0061;
(f) char c = 97;

3.26) Which of the following are valid for declaring and initialising a boolean variable?
(a) boolean b = True;
(b) boolean b = 0;
(c) boolean b = 1 < 2;
(d) boolean b = true?false:true;
(e) boolean b = true;

3.27) Select the valid declarations.
(a) int i = 16.0;
(b) byte b = 16.0;
(c) float f = 16.0;
(d) double d = 16.0;
(e) char \u0063 = '\u0063';

3.28) What is the result of attempting to compile and run this code

```
public class Test{
  int i;
  Test(){
  i++;
  }
  public static void main(String[] args){
  Test t = new Test();
  System.out.println("value = " t.i);
  }
}
```

(a) The code will compile and run printing out value = 1
(b) The code will compile and run printing out value = 0
(c) The code will fail to compile becuse i has not been initialised
(d) Run time error

3.29) What is the result of attempting to compile and run the following

```
public class Test{
   int arr[] = new int[8];
   public static void main(String[] args){
    Test t = new Test();
   System.out.println(t.arr[7]);
    }
}
```

(a) Compiler error
(b) Runtime exception
(c) Compiles and runs printing out 0

3.30) What is the result of trying to compile and run this

```
public class Test{
    public static void main(String[] args){
int a;
       Test t = new Test();
```

```
                for(int j=0; j < 2 ; j++){
                    for(int i=0; i <4 ; i++){
                     a = j;
                     }
                System.out.println(i);
                }
              }
          }
```

(a) Compiles without error and runs printing out 1 twice.
(b) Compiles without error and runs printing out 0 followed by 1 .
(c) Compiles without error and runs printing out 0 twice.
(d) Runtime error
(e) Does not compile

3.31) Given the following declaration, which expression returns the size of the array, assuming the array has been initialized?

```
int[] array;
```

Select the one correct answer.
(a) array[].length()
(b) array.length()
(c) array[].length
(d) array.length
(e) array[].size()
(f) array.size()

3.32) Is it possible to create arrays of length zero? Select the one correct answer.
(a) Yes, you can create arrays of any type with length zero.
(b) Yes, but only for primitive data types.
(c) Yes, but only for arrays of object references.
(d) No, you cannot create zero-length arrays, but the main() method may be passed a zero-length array of String references when no program arguments are specified.
(e) No, it is not possible to create arrays of length zero in Java.

3.33) Which one of the following array declaration statements is not legal? Select the one correct answer.
(a) int []a[] = new int [4][4];
(b) int a[][] = new int [4][4];
(c) int a[][] = new int [][4];
(d) int []a[] = new int [4][];
(e) int [][]a = new int [4][4];

3.34) Which of these array declaration statements are not legal? Select the two correct answers.
(a) int[] i[] = { { 1, 2 }, { 1 }, {}, { 1, 2, 3 } };
(b) int i[] = new int[2] {1, 2};
(c) int i[][] = new int[][] { {1, 2, 3}, {4, 5, 6} };
(d) int i[][] = { { 1, 2 }, new int[2] };
(e) int i[4] = { 1, 2, 3, 4 };

3.35) What would be the result of attempting to compile and run the following program?

```
// Filename: MyClass.java
class MyClass {
      public static void main(String[] args) {
            int size = 20;
            int[] arr = new int[ size ];
            for (int i = 0; i < size; ++i) {
                  System.out.println(arr[i]);
            }
      }
}
```

Select the one correct answer.
(a) The code will fail to compile because the array typei nt[] is incorrect.
(b) The program will compile, but will throw an ArrayIndexOutOfBoundsException when run.

(c) The program will compile and run without error, but will produce no output.
(d) The program will compile and run without error and will print the numbers 0 through 19.
(e) The program will compile and run without error and will prin0t twenty times.
(f) The program will compile and run without error and will prinnt ull twenty times.

3.36) Given the following program, which statement is true?

```
class MyClass {
      public static void main(String[] args) {
            String[] numbers = { "one", "two", "three", "four" };
            if (args.length == 0) {
                  System.out.println("no arguments");
            } else {
                  System.out.println(numbers[ args.length ] + " arguments");
            }
      }
}
```

Select the one correct answer.
(a) The program will fail to compile.
(b) The program will throw a NullPointerException when run with zero program arguments.
(c) The program will print "no arguments" and "two arguments" when called with zero and three program arguments, respectively.
(d) The program will print "no arguments" and "three arguments" when called with zero and three program arguments, respectively.
(e) The program will print "no arguments" and "four arguments" when called with zero and three program arguments, respectively.
(f) The program will print "one arguments" and "four arguments" when called with zero and three program arguments, respectively.

3.37) What would be the result of trying to compile and run the following program?

```
public class DefaultValuesTest {
      int[] ia = new int[1];
      boolean b;
      int i;
      Object o;
      public static void main(String[] args) {
            DefaultValuesTest instance = new DefaultValuesTest();
            instance.print();
      }
      public void print() {
            System.out.println(ia[0] + " " + b + " " + i + " " + o);
      }
}
```

Select the one correct answer.
(a) The program will fail to compile because of uninitialized variables.
(b) The program will throw a java.lang.NullPointerException when run.
(c) The program will print "0 false NaN null".
(d) The program will print "0 false 0 null.
(e) The program will print "null 0 0 null.
(f) The program will print "null false 0 null.

3.38) Which one of these is a valid method declaration? Select the one correct answer.
(a) void method1 { /* ... */ }
(b) void method2() { /* ... */ }
(c) void method3(void) { /* ... */ }
(d) method4() { /* ... */ }
(e) method5(void) { /* ... */ }

3.39) Given the following code, which statements can be placed at the indicated position without causing compilation errors?

```
      public class ThisUsage {
            int planets;
```

```
        static int suns;
        public void gaze() {
        int i;
        // ... insert statements here ...
        }
}
```

Select the three correct answers.

(a) i = this.planets;
(b) i = this.suns;
(c) this = new ThisUsage();
(d) this.i = 4;
(e) this.suns = planets;

3.40) Given the following pairs of method declarations, which statements are true?

```
void fly(int distance) {}
int fly(int time, int speed) { return time*speed; }
void fall(int time) {}
int fall(int distance) { return distance; }
void glide(int time) {}
void Glide(int time) {}
```

Select the two correct answers.

a. The first pair of methods will compile correctly and overload the method namef ly.
b. The second pair of methods will compile correctly and overload the method namef all.
c. The third pair of methods will compile correctly and overload the method nameg lide.
d. The second pair of methods will not compile correctly.
e. The third pair of methods will not compile correctly.

3.41) Given a class named Book, which one of these is a valid constructor declaration for the class? Select the one correct answer.

a. Book(Book b) {}
b. Book Book() {}
c. private final Book() {}
d. void Book() {}
e. public static void Book(String[] args) {}
f. abstract Book() {}

3.42) Which statements are true? Select the two correct answers.

a. All classes must define a constructor.
b. A constructor can be declared private.
c. A constructor can return a value.
d. A constructor must initialize all the fields of a class.
e. A constructor can access the non-static members of a class.

3.43) What will be the result of attempting to compile the following program?

```
public class MyClass {
        long var;
        public void MyClass(long param) { var = param; } // (1)
        public static void main(String[] args) {
                MyClass a, b;
                a = new MyClass(); // (2)
                b = new MyClass(5); // (3)
        }
}
```

Select the one correct answer.

a. A compilation error will occur at (1), since constructors cannot specify a return value.
b. A compilation error will occur at (2), since the class does not have a default constructor.
c. A compilation error will occur at (3), since the class does not have a constructor which takes one argument of type int.
d. The program will compile correctly.

3.44) Given the following class, which of these are valid ways of referring to the class from outside of the package

```
net.basemaster?
package net.basemaster;
public class Base {
// ...
}
```

Select the two correct answers.

a. By simply referring to the class as Base.
b. By simply referring to the class as basemaster.Base.
c. By simply referring to the class as net.basemaster.Base.
d. By importing with net.basemaster.* and referring to the class as Base.
e. By importing with net.* and referring to the class as basemaster.Base.

3.45) Which one of the following class definitions is a valid definition of a class that cannot be instantiated?
Select the one correct answer.

a.class Ghost {
abstract void haunt();
}
b.abstract class Ghost {
void haunt();
}
c. abstract class Ghost {
void haunt() {};
}
d. abstract Ghost {
abstract void haunt();
}
e. static class Ghost {
abstract haunt();
}

3.46) Which one of the following class definitions is a valid definition of a class that cannot be extended?
Select the one correct answer.

a. class Link { }
b. abstract class Link { }
c. native class Link { }
d. static class Link { }
e. final class Link { }
f. private class Link { }
g. abstract final class Link { }

3.47) Given the following definition of a class, which fields are accessible from outside the package

```
com.corporation.project?
package com.corporation.project;
public class MyClass {
        int i;
        public int j;
        protected int k;
        private int l;
}
```

Select the two correct answers.

a. Field i is accessible in all classes in other packages.
b. Field j is accessible in all classes in other packages.
c. Field k is accessible in all classes in other packages.
d. Field k is accessible in subclasses only in other packages.
e. Field l is accessible in all classes in other packages.
f. Field l is accessible in subclasses only in other packages.

3.48) How restrictive is the default accessibility compared to public, protected, and private accessibility? Select the one correct answer.

a. Less restrictive than public.
b. More restrictive than public, but less restrictive thanp rotected.
c. More restrictive than protected, but less restrictive thanp rivate.
d. More restrictive than private.
e. Less restrictive than protected from within a package, and more restrictive thanp rotected from outside a package.

3.49) Which statement is true about accessibility of members? Select the one correct answer.

a. Private members are always accessible from within the same package.
b. Private members can only be accessed by code from within the class of the member.
c. A member with default accessibility can be accessed by any subclass of the class in which it is defined.
d. Private members cannot be accessed at all.
e. Package/default accessibility for a member can be declared using the keywordd efault.

3.50) Which statements are true about the use of modifiers? Select the two correct answers.

a. If no accessibility modifier (public, protected, and private) is specified for a member declaration, the member is only accessible for classes in the package of its class and subclasses of its class anywhere.
b. You cannot specify accessibility of local variables. They are only accessible within the block in which they are declared.
c. Subclasses of a class must reside in the same package as the class they extend.
d. Local variables can be declared static.
e. Objects themselves do not have any accessibility modifiers, only the object references do.

3.51) Given the following source code, which comment line can be uncommented without introducing errors?

```
abstract class MyClass {
      abstract void f();
      final void g() {}
      // final void h() {} // (1)
      protected static int i;
      private int j;
}
      final class MyOtherClass extends MyClass {
            // MyOtherClass(int n) { m = n; } // (2)
            public static void main(String[] args) {
            MyClass mc = new MyOtherClass();
      }
      void f() {}
      void h() {}
      // void k() { i++; } // (3)
      // void l() { j++; } // (4)
      int m;
}
```

Select the one correct answer.

a. final void h() {} // (1)
b. MyOtherClass(int n) { m = n; } // (2)
c. void k() { i++; } // (3)
d. void l() { j++; } // (4)

3.52) What would be the result of attempting to compile and run the following program?

```
class MyClass {
      static MyClass ref;
      String[] arguments;
      public static void main(String[] args) {
            ref = new MyClass();
            ref.func(args);
      }
```

```
	public void func(String[] args) {
		ref.arguments = args;
	}
}
```

Select the one correct answer.

a. The program will fail to compile, since the static method main() cannot have a call to the non-static method func().
b. The program will fail to compile, since the non-static method func() cannot access the static variable ref.
c. The program will fail to compile, since the argument args passed to the static method main() cannot be passed on to the non-static method func().
d. The program will fail to compile, since the method func() cannot assign the value of the static variable ref to the non-static variable arguments.
e. The program will compile, but will throw an exception when run.
f. The program will compile and run successfully.

3.53) Given the following member declarations, which statement is true?

```
	int a; // (1)
	static int a; // (2)
	int f() { return a; } // (3)
	static int f() { return a; } // (4)
```

Select the one correct answer.

a. Declarations (1) and (3) cannot occur in the same class definition.
b. Declarations (2) and (4) cannot occur in the same class definition.
c. Declarations (1) and (4) cannot occur in the same class definition.
d. Declarations (2) and (3) cannot occur in the same class definition.

3.54) Which statement is true? Select the one correct answer.

a. A static method can call other non-static methods in the same class by using thet his keyword.
b. A class may contain both static and non-static variables and both static and non-static methods.
c. Each object of a class has its own instance of each static variable.
d. Instance methods may access local variables of static methods.
e. All methods in a class are implicitly passed at his parameter when called.

3.55) What, if anything, is wrong with the following code?

```
	abstract class MyClass {
		transient int j;
		synchronized int k;
		final void MyClass() {}
		static void f() {}
	}
```

Select the one correct answer.

a. The class MyClass cannot be declared abstract.
b. The field j cannot be declared transient.
c. The field k cannot be declared synchronized.
d. The method MyClass() cannot be declared final.
e. The method f() cannot be declared static.
f. Nothing is wrong with the code; it will compile without errors.

3.56) Which one of these is not a legal member declaration within a class? Select the one correct answer.

a. static int a;
b. final Object[] fudge = { null };
c. abstract int t;
d. native void sneeze();
e. final static private double PI = 3.14159265358979323846;

3.57) Which statements are true about modifiers? Select the two correct answers.

a. Abstract classes can contain final methods.
b. Fields can be declared native.
c. Non-abstract methods can be declared in abstract classes.
d. Classes can be declared native.

e. Abstract classes can be declared final.

3.58) Which statement is true? Select the one correct answer.

a. Transient fields will not be saved during serialization.
b. Constructors can be declared abstract.
c. The initial state of an array object constructed with the statement int[] a = new int[10] will depend on whether the array variable a is a local variable or a field.
d. A subclass of a class with ana bstract method must provide an implementation for thea bstract method.
e. Only static methods can access static members.

Declarations & Access Control Answers

3.1) (c)
The [] notation can be placed both before and after the variable name in an array declaration. Multidimensional arrays are created by creating arrays that can contain references to other arrays. The statement new int[4][] will create an array of length 4, which can contain references to arrays of int values. The statement new int[4][4] will create the same array, but will also create four arrays, each containing four int values. References to each of these arrays are stored in the first array. The statement int [][4] will not work, because the dimensions must be created from left to right. Extra spaces are not significant.

3.2) (b), (e)
The size of the array cannot be specified as in (b), (e). The size of the array is given implicitly by the initialization code. The sizer of the array is never given during the declaration of an array reference. The size of an array is always associated with the array instance, not the array reference.

3.3) (a), (b), (e)
Non-static methods have an implicit this object reference. The this reference is not a normal reference variable that can be changed in the way attempted by statement (c). The this reference can be used to refer to both object and class members within the current context. However, it cannot be used to refer to local variables in the way attempted by statement (d).

3.4) (a), (d)
The first and third pairs of methods will compile correctly. The second pair of methods will not compile correctly, since their method signatures do not differ and the compiler has therefore no way of diffrentiating between the two methods. Note that return type and the names of the parameters are not a part of the method signatures. Both methods in the first pair named fly and therefore overload this method name. The methods in pair three do not overload the method name glide, since only one method has the name. The method name Glide is distinct from the method name glide, as identifiers in Java are case-sensitive.

3.5) (a)
A constructor does not declare any return type, not even void. A constructor cannot be final, static or abstract.

3.6) (b), (e)
A constructor can be declared private, but this means that this constructor can only be used directly within the class. Constructors need not initialize all the member variables in a class. A member variable will be assigned a default value if not explicitly initialized. A constructor is non-static, and as such it can access directly both the static and non-static members of the class.

3.7) (c)
A compilation error will be encountered in (3), since the class does not have a constructor accepting a single argument of type int. The declaration at (1) declares a method, not a constructor, since it have a return type. The method happens to have the same name as the class, but that is irrelevant. The class has an implicit default constructor since the class contains no constructor declarations. This allows the instantiation at (2) to work.

3.8) (c), (d)
A class or interface name can be referred to by using either its fully qualified name or its simple name. Using the fully qualified name will always work, but in order to use the simple name it has to be imported. By importing net.basemaster.* all the type names from the package net.basemaster will be imported and can now be referred to using simple names. Importing net.* will not import the subpackage basemaster.

3.9) (C)
A class is uninstantiable if the class is declared abstract. The declarartion of an abstract method cannot provide an implementation. The declaration of a non-abstract method must provide an implemenation.

If any method in a class is declared abstract, then the whole class must be declared abstract. Definition (d) is not valid, since it omits the class keyword.

3.10) (b), (e)

You cannot specify visibility of local variables. They are accessible only within the block in which they are declared. Objects themselves do not have any visibility, only the references to the object. If no visibility modifier(public, protected or private) is given in the member declaration of a class, the member is only accessible to classes in the same package. A class does not have access to members of a superclass with default accessibility, unless both classes are in the same package. Inheritance has no consequence with respect to accessing members wit hdefaul;t accessibility. Local variables acannot be declared static and cannot be given an accessibility modifier.

3.11) (c)

The line "void k() { i++; }" can be reinserted without introducing errors. Reinserting line (1) will cause the compilation to fail, since MyOtherClass will try to override a final method. Reinserting line(2) will fail since MyOtherClass will no longer have a default constructor. The main() method needs a constructor that takes zero arguments. Reinserting line (3) will work without any problems, but reinserting line (4) will fail, since the method will try to access a private member of the superclass.

3.12) (b)

The keyword this can be only used in non-static methods. Only one instance of each static member variable of a class is created. This instance is shared among all objects of the class. Local variables are only accessible within the local scope, regardless of whether the local scope is defined within a static method.

3.13) (a)

The transient keyword signifies that the variables should not be stored when the object are serialized. Constructors cannot be declared abstract. Elements in an unitialized array object get the default value corresponding to the type of the elements. Whether the reference variable pointing to the array object is a local or a member variable does not matter. Abstract methods from a superclass need not be implemented by abstract subclass.

3.14) (a), (b), (d)

The keyword transient is easy to forget as is not frequently used, and it is considered in the access modifiers because it modifies the behaviour of the object.

3.15) (a), (c)

3.16) (d)

It would cause a run time error if you had a call to amethod though.

3.17) (a)

A top level (non nested) class cannot be private.

3.18) (d)

3.19) (a)

You will get an error saying something like "Cant make a static reference to a non static variable". Note that the main method is static. Even if main was not static the array argv is local to the main method and would thus not be visible within amethod.

3.20) (b)

Interface methods can't be native, static, synchronized, final, private, or protected.

3.21) (a)

All interface methods are implicitly public.
Will fail compilation becuse the access modifier for aMethod() is being made more restrictive.

3.22) (b)

Local variables cannot be declared static.

3.23) (b)
A variable declared in a static initialiser is not accessible outside its enclosing block.

3.24) (b), (d), (e), (f)

3.25) (a), (b), (c), (f)

3.26) (c), (d),(e)

3.27) (d), (e)
16.0 is a double type literal hence it becomes a widening assignment for a byte, int or float.

3.28) (a)
Numeric instance variables are always automatically initialised to its default value. So you can use them without explicitly initialising them.

3.29) (c)
Numeric instance variables are always automatically initialised to its default value. In the case of arrays each element of the array is initialised to its default value.

3.30) (e)
The variable i is not accessible outside the enclosing block of the inner loop.

3.31) (d)
In Java, arrays are objects. Each array object has afi nal field named length that stores the size of the array.

3.32) (a)
Java allows arrays of length zero. Such an array is passed as an argument to the main() method when a Java program is run without any program arguments.

3.33) (c)
The [] notation can be placed both before and after the variable name in an array declaration. Multidimensional arrays are created by constructing arrays that can contain references to other arrays. The expression new int[4][] will create an array of length 4, which can contain references to arrays of int values. The expression new int[4][4] will create the same array, but will in addition create four more arrays, each containing four int values. References to each of these arrays are stored in the first array. The expression int[][4] will not work, because the arrays for the dimensions must be created from left to right.

3.34) (b) and (e)
The size of the array cannot be specified as in (b) and (e). The size of the array is given implicitly by the initialization code. The size of the array is never specified in the declaration of an array reference. The size of an array is always associated with the array instance, not the array reference.

3.35 (e)
The array declaration is valid and will declare and initialize an array of length 20 containing int values. All the values of the array are initialized to their default value of 0. The for loop will print all the values in the array, that is, it will prin0t twenty times.

3.36) (e)
The program will type "no arguments" and "four arguments" when called with 0 and 3 arguments, respectively. When the program is called with no arguments, the args array will be of length zero. The program will in this case type "no arguments". When the program is called with three arguments, the args array will have length 3. Using the index 3 on the numbers array will retrieve the string "four", because the start index is 0.

3.37) (d)
The program will print "0 false 0 null" when run. All the instance variables, including the array element, will be initialized to their default values. When concatenated with a string, the values are converted to their

string representation. Notice that the null pointer is converted to the string "null" rather than throwing a NullPointerException.

3.38) (b)
Only (b) is a valid method declaration. Methods must specify a return type or are declared void. This makes (d) and (e) invalid. Methods must specify a list of zero or more comma-separated parameters delimited by ( ). The keyword void is not a valid type for a parameter. This makes (a) and (c) invalid.

3.39) (a), (b), and (e)
Non-static methods have an implicit this object reference. The this reference cannot be changed, as shown in (c). The this reference can be used in a non-static context to refer to both instance and static members. However, it cannot be used to refer to local variables, as shown in (d).

3.40 (a) and (d)
The first and the third pairs of methods will compile correctly. The second pair of methods will not compile correctly, since their method signatures do not differ. The compiler has no way of differentiating between the two methods. Note that return type and the names of the parameters are not a part of the method signatures. Both methods in the first pair are named fly and, therefore, overload this method name. The methods in pair three do not overload the method name glide, since only one method has that name. The method named Glide is distinct from the method named glide, as identifiers in Java are case sensitive.

3.41) (a)
A constructor cannot specify any return type, not even void. A constructor cannot be final, static or abstract.

3.42) (b) and (e)
A constructor can be declared private, but this means that this constructor can only be used within the class. Constructors need not initialize all the fields of the class. A field will be assigned a default value if not explicitly initialized. A constructor is non-static, and as such it can directly access both the static and non-static members of the class.

3.43) (c)
A compilation error will occur at (3), since the class does not have a constructor accepting a single argument of type int. The declaration at (1) declares a method, not a constructor, since it is declared as void. The method happens to have the same name as the class, but that is irrelevant. The class has an implicit default constructor since the class contains no constructor declarations. This constructor is invoked to create a MyClass object at (2).

3.44) (c) and (d)
A class or interface name can be referred to by using either its fully qualified name or its simple name. Using the fully qualified name will always work, but in order to use the simple name it has to be imported. By importing net.basemaster.* all the type names from the package net.basemaster will be imported and can now be referred to using simple names. Importing net.* will not import the subpackage basemaster.

3.45) (c)
A class is uninstantiable if the class is declared abstract. The declaration of an abstract method cannot provide an implementation. The declaration of a non-abstract method must provide an implementation. If any method in a class is declared abstract, then the class must be declared abstract. Definition (d) is not valid since it omits the class keyword.

3.46) (e)
A class can be extended unless it is declared final. For classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass. A nested static class, (d), can be extended. A private member class, (f), can also be extended. The keyword native can only be used for methods, not for classes and fields.

3.47) (b) and (d)
Outside the package, member j is accessible to any class, whereas member k is only accessible to subclasses of MyClass. Field i has package accessibility and is only accessible by classes inside the package. Fieldj has public accessibility and is accessible from anywhere. Field k has protected accessibility and is accessible from

any class inside the package and from subclasses anywhere. Field l has private accessibility and is only accessible within its own class.

3.48) (c)
The default accessibility for members is more restrictive than protected accessibility, but less restrictive than private. Members with default accessibility are only accessible within the class itself and from classes in the same package. Protected members are in addition accessible from subclasses anywhere. Members with private accessibility are only accessible within the class itself.

3.49) (b)
A private member is only accessible by code from within the class of the member. If no accessibility modifier has been specified, a member has default accessibility, also known as package accessibility. The keyword default is not an accessibility modifier, and its only use is as a label in a switch statement. Members with package accessibility are only accessible from classes in the same package. Subclasses outside the package cannot access members with default accessibility.

3.50) (b) and (e)
You cannot specify accessibility of local variables. They are accessible only within the block in which they are declared. Objects themselves do not have any accessibility, only references to objects do. If no accessibility modifier (public, protected, or private) is given in the member declaration of a class, the member is only accessible to classes in the same package. A class does not have access to members with default accessibility declared in a superclass, unless both classes are in the same package. Inheritance has no consequence with respect to accessing members with default accessibility in the same package. Local variables cannot be declared static or given an accessibility modifier.

3.51) (c)
The line void k() { i++; } can be re-inserted without introducing errors. Re-inserting line (1) will cause the compilation to fail, since MyOtherClass will try to override a final method. Re-inserting line (2) will fail, since MyOtherClass will no longer have a default constructor. The main() method needs to call the default constructor. Re-inserting line (3) will work without any problems, but re-inserting line (4) will fail, since the method will try to access a private member of the superclass.

3.52) (f)
An object reference is needed to access non-static members. Static methods do not have the implicit object reference this, and must always supply an explicit object reference when referring to non-static members. The static method main() refers legally to the non-static method func() using the reference variable ref. Static members are accessible both from static and non-static methods, using their simple names.

3.53 (c)
Local variables can have the same name as member variables. The local variables will simply shadow the member variables with the same names. Declaration (4) defines a static method that tries to access a variable named a, which is not locally declared. Since the method is static, this access will only be valid if variable a is declared static within the class. Therefore, declarations (1) and (4) cannot occur in the same class definition, while declarations (2) and (4) can.

3.54) (b)
The keyword this can only be used in non-static code, like in non-static methods. Only one occurrence of each static variable of a class is created. This occurrence is shared among all the objects of the class (or for that matter, by other clients). Local variables are only accessible within the local scope, regardless of whether the local scope is defined within a static context.

3.55) (c)
The variable k cannot be declared synchronized. Only methods and code blocks can be synchronized.

3.56 (c)
The declaration abstract int t; is not legal. Keywords static and final are valid modifiers for both field and method declarations. The modifiers abstract and native are only valid for methods.

3.57) (a) and (c)

Abstract classes can contain both final methods and non-abstract methods. Non-abstract classes cannot, however, contain abstract methods. Nor can abstract classes be final. Only methods can be declared native.

3.58) (a)
The transient keyword signifies that the fields should not be stored when objects are serialized. Constructors cannot be declared abstract. When an array object is created, as in (c), the elements in the array object are assigned the default value corresponding to the type of the elements. Whether the reference variable denoting the array object is a local or a member variable is irrelevant. Abstract methods from a superclass need not be implemented by a subclass. The subclass must then be declared abstract.

Flow Control, Exception Handling & Assertions Mock Questions

4.1) The following method is designed to convert an input string to a floating point number while detecting a bad format.

```
public boolean strCvt(String s) {
    try {
        factor = Float.valueOf(s).floatValue();
        return true;
    } catch (NumberFormatException e) {
        System.out.println("Bad number " + s);
        factor = Float.NaN;
    } finally {
        System.out.println("Finally");
    }
    return false;
}
```

Which of the following descriptions of the results of various inputs to the method are correct?

(a) Input = "0.234" – Result: factor = 0.234, "Finally" is printed, true is returned.
(b) Input = "0.234" – Result: factor = 0.234, "Finally" is printed, false is returned.
(c) Input = null – Result: factor = NaN, "Finally" is printed, false is returned.
(d) Input = null – Result: factor = unchanged, "Finally" is printed, NullPointerException is thrown.

4.2) Here is the hierarchy of exceptions related to array index and string index errors:

Exception
+-- RuntimeException
+-- IndexOutOfBoundsException
+-- ArrayIndexOutOfBoundsException
+-- StringIndexOutOfBoundsException

Suppose you had a method *X* that could throw both array index and string index exceptions. Assuming that *X* does not have any try-catch statements, which of the following statements are correct?

(a) The declaration for *X* must include "throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException".
(b) If a method calling *X* catches IndexOutOfBoundsException, both array and string index exceptions will be caught.
(c) If the declaration for *X* includes "throws IndexOutOfBoundsException", any calling method must use a try-catch block.
(d) The declaration for *X* does not have to mention exceptions.

4.3) Which will be the first line to cause an error in the following code?

Select one correct answer.

```
1  class Char
2  {
3     public static void main(String arg[])
4     {
5        while(false)
6        {
7            System.out.println("Hello");
8        }
9        while(false)
10       {
11       }
12       do;
13       while(false);
14       do
15       {
16           ;
17       }
18       while(false);
19    }
20 }
```

(a) Line no. 5
(b) Line no. 9
(c) Line no. 12
(d) Line no. 16

4.4) What will be the result of compiling and running the given program?
Select one correct answer.

```
1   public class exception
2   {
3      public static void main(String args[])
4      {
5          System.out.println("A");
6          try
7          {
8          }
9          catch(java.io.IOException t)
10         {
11            System.out.println("B");
12         }
13         System.out.println("C");
14     }
15  }
```

(a) Compile time error.
(b) Program compiles correctly and prints "A" when executed.
(c) Program compiles correctly and prints "A" and "C" when executed.
(d) Run time error.

4.5) What will be the result of compiling and running the given program?
Select one correct answer.

```
1   public class exception
2   {
3      public static void main(String args[])
4      {
5          System.out.println("A");
6          try
7          {
8              return;
9          }
10         catch(Exception e)
11         {
12            System.out.println("B");
13         }
14         System.out.println("C");
15     }
16  }
```

(a) Compile time error in line no. 8 as main() method is declared void.
(b) Program compiles correctly and prints "A" when executed.
(c) Program compiles correctly and prints "A" and "C" when executed.
(d) Compile time error at line no.14 due to statement not reached.

4.6) What will be the result of compiling and running the given program?
Select one correct answer.

```
1   public class exception
2   {
3      public static void main(String args[])
4      {
5          System.out.println("A");
6          try
7          {
8              return;
9          }
10         catch(Exception e)
```

```
11          {
12             System.out.println("B");
13          }
14          finally
15          {
16             System.out.println("C");
17          }
18      }
19  }
```

(a) Compile time error in line no. 8 as main() method is declared void.
(b) Program compiles correctly and prints "A" when executed.
(c) Program compiles correctly and prints "A" and "C" when executed.
(d) Program compiles correctly and prints "A","B" and "C" when executed.

4.7) What will be the result of compiling and running the given program?
Select one correct answer.

```
1   public class exception
2   {
3      public static void main(String args[])
4      {
5          System.out.println("A");
6          try
7          {
8             System.out.println("B");
9             System.exit(0);
9          }
10         catch(Exception e)
11         {
12            System.out.println("C");
13         }
14         finally
15         {
16            System.out.println("D");
17         }
18      }
19  }
```

(a) Program compiles correctly and prints "A" when executed.
(b) Program compiles correctly and prints "A" and "B" when executed.
(c) Program compiles correctly and prints "A" and "C" when executed.
(d) Program compiles correctly and prints "A","B" and "C" when executed.

4.8) What is the result of attempting to compile and run this code ?

```
public class Test {
 public static void main(String[] args){
 int j = 0;
     for(; j < 3; j++){
     if (j==1) break out;
     System.out.print(j + "\n");
     }
 out:{System.out.println("bye");}
 }
}
```

(a) The code will fail to compile.
(b) The code will run with no out put
(c) This will run and print 0, 1 , 2 and "bye"
(d) This will run and print 1 and "bye"

4.9) What is the result of attempting to compile and run this code ?

```
class A extends Exception{}
class B extends A{}
```

```
class C extends B{}
public class Test {
   static void aMethod() throws C{ throw new C(); }
   public static void main(String[] args){
   int x = 10;
    try { aMethod(); }
    catch(A e) { System.out.println("Error A");}
    catch(B e) { System.out.println("Error B");}
   }
}
```

(a) Compiler error
(b) It will print "Error A"
(c) It will print "Error B"
(d) The exception will go uncaught by both catch blocks

4.10) Will the print line statement execute here?

```
while(true?true:false)
 {
  System.out.println("hello");
  break;
 }
```

(a) Yes
(b) No

4.11) What is the result of trying to compile and run this?

```
public class Test {
  public static void main(String[] args){
    for(int i=0; i < 2; i++)
    {
    continue;
    System.out.println("Hello world");
    }
  }
}
```

(a) Prints "Hello world" once
(b) Prints "Hello world" twice
(c) Compiler error
(d) Runs without any output

4.12) Consider this code.

```
 public class Test {
   public static void main(String[] args){
      for(int j = 0; j < 1 ; j++)
      {
    if (j < 1) continue innerLoop;
innerLoop:      for(int i = 0; i < 2; i++)
      {
      System.out.println("Hello world");
      }
    }
   }
 }
```

What is the result of attempting to compile and run this.
(a) The code will not compile.
(b) It will run and print "Hello world" twice.
(c) It will run and print "Hello world" once.
(d) It will run and print "Hello world" thrice.
(e) It will run with no output

4.13) What will the output be ?

```
  public static void main(String[] args){
  char c = '\u0042';
```

```
switch(c) {
default:
System.out.println("Default");
case 'A':
System.out.println("A");
case 'B':
System.out.println("B");
case 'C':
System.out.println("C");
}
}
```

(a) Prints - Default , A , B , C
(b) Prints - A
(c) Prints - B , C
(d) Prints - A, B , C , Default
(e) Prints - Default

4.14) What wil be printed out when this method runs ?

```
void getCount(){
  int counter = 0;
   for (int i=10; i>0; i--) {
        int j = 0;
        while (j > 10) {
           if (j > i) break;
             counter++;
             j++;
           }
   }
   System.out.println(counter);
}
```

(a) 64
(b) 53
(c) 76
(d) 0

4.15) Is this code legal ?

```
class ExceptionA extends Exception {}
class ExceptionB extends ExceptionA {}
public class Test{
    void thrower() throws ExceptionB{
    throw new ExceptionB();
    }
    public static void main(String[] args){
     Test t = new Test();
     try{t.thrower();}
     catch(ExceptionA e) {}
     catch(ExceptionB e) {}
    }
}
```

(a) Yes
(b) No

4.16) Is this legal ?

```
class ExceptionA extends Exception {}
class ExceptionB extends ExceptionA {}
public class Test{
    void thrower() throws ExceptionA{
     throw new ExceptionA();
    }
    public static void main(String[] args){
       Test t = new Test();
       try{t.thrower();}
```

```
                catch(ExceptionB e) {}
              }
            }
```

(a) Yes
(b) No

4.17) Is this legal ?

```
class ExceptionA extends Exception {}
class ExceptionB extends ExceptionA {}
class A{
  void thrower() throws ExceptionA{
  throw new ExceptionA();
   }
}
public class B extends A{
   void thrower() throws ExceptionB{
      throw new ExceptionB();
   }
}
```

(a) Yes
(b) No

4.18) What will be the result of attempting to compile and run the following class?

```
public class IfTest {
      public static void main(String[] args) {
      if (true)
      if (false)
            System.out.println("a");
      else
            System.out.println("b");
      }
}
```

Select the one correct answer.

a. The code will fail to compile because the syntax of thei f statement is incorrect.
b. The code will fail to compile because the compiler will not be able to determine which if statement the else clause belongs to.
c. The code will compile correctly and display the lettera when run.
d. The code will compile correctly and display the letterb when run.
e. The code will compile correctly, but will not display any output.

4.19) Which statements are true? Select the three correct answers.

a. The conditional expression in an if statement can have method calls.
b. If a and b are of type boolean, the expression (a = b) can be the conditional expression of an if statement.
c. An if statement can have either an if clause or an else clause.
d. The statement if (false) ; else ; is illegal.
e. Only expressions which evaluate to ab oolean value can be used as the condition in ani f statement.

4.20) What, if anything, is wrong with the following code?

```
void test(int x) {
      switch (x) {
            case 1:
            case 2:
            case 0:
            default:
            case 4:
      }
}
```

Select the one correct answer.

a. The variable x does not have the right type for as witch expression.
b. The case label 0 must precede case label 1.

c. Each case section must end with ab reak statement.
d. The default label must be the last label in thes witch statement.
e. The body of the switch statement must contain at least one statement.
f. There is nothing wrong with the code.

4.21) Which of these combinations of switch expression types and case label value types are legal within a switch statement? Select the one correct answer.
a. switch expression of type int and case label value of type char.
b. switch expression of type float and case label value of type int.
c. switch expression of type byte and case label value of type float.
d. switch expression of type char and case label value of type long.
e. switch expression of type boolean and case value of type boolean.

4.22) What will be the result of attempting to compile and run the following code?

```
class MyClass {
      public static void main(String[] args) {
            boolean b = false;
            int i = 1;
            do {
                  i++;
                  b = ! b;
            } while (b);
            System.out.println(i);
      }
}
```

Select the one correct answer.
a. The code will fail to compile, since b is an invalid conditional expression for the do-while statement.
b. The code will fail to compile, since the assignment b = ! b is not allowed.
c. The code will compile without error and will print1 when run.
d. The code will compile without error and will print2 when run.
e. The code will compile without error and will print3 when run.

4.23) What will be the output when running the following program?

```
public class MyClass {
      public static void main(String[] args) {
            int i=0;
            int j;
            for (j=0; j<10; ++j) { i++; }
                  System.out.println(i + " " + j);
      }
}
```

Select the two correct answers.
a. The first number printed will be 9.
b. The first number printed will be 10.
c. The first number printed will be 11.
d. The second number printed will be 9.
e. The second number printed will be 10.
f. The second number printed will be 11.

4.24) Which one of these for statements is valid? Select the one correct answer.
a. int j=10; for (int i=0, j+=90; i<j; i++) { j--; }
b. for (int i=10; i=0; i--) {}
c. for (int i=0, j=100; i<j; i++, --j) {;}
d. int i, j; for (j=100; i<j; j--) { i += 2; }
e. int i=100; for ((i>0); i--) {}

4.25) What will be the result of attempting to compile and run the following program?

```
class MyClass {
      public static void main(String[] args) {
            int i = 0;
```

```
                for ( ; i<10; i++) ; // (1)
                for (i=0; ; i++) break; // (2)
                for (i=0; i<10; ) i++; // (3)
                for ( ; ; ) ; // (4)
            }
        }
```

Select the one correct answer.
a. The code will fail to compile, since the for statement (1) is missing the expression in the first section.
b. The code will fail to compile, since the for statement (2) is missing the expression in the middle section.
c. The code will fail to compile, since the for statement (3) is missing the expression in the last section.
d. The code will fail to compile, since the for statement (4) is invalid.
e. The code will compile without error, and the program will run and terminate without any output.
f. The code will compile without error, but will never terminate when run.

4.26) Which statements are valid when occurring on their own?> Select the three correct answers.
a. while () break;
b. do { break; } while (true);
c. if (true) { break; }
d. switch (1) { default: break; }
e. for (;true;) break;

4.27) Given the following code fragment, which of the following lines will be a part of the output?

```
        outer:
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 2; j++) {
                if (i == j) {
                continue outer;
            }
                System.out.println("i=" + i + ", j=" + j);
            }
        }
```

Select the two correct answers.
a. i=1, j=0
b. i=0, j=1
c. i=1, j=2
d. i=2, j=1
e. i=2, j=2
f. i=3, j=3
g. i=3, j=2

4.28) What will be the result of attempting to compile and run the following code?

```
class MyClass {
        public static void main(String[] args) {
            for (int i = 0; i<10; i++) {
                switch(i) {
                case 0:
                System.out.println(i);
            }
            if (i) {
                System.out.println(i);
            }
            }
        }
}
```

Select the one correct answer.
a. The code will fail to compile, owing to an illegal switch expression in the switch statement.
b. The code will fail to compile, owing to an illegal conditional expression in the if statement.
c. The code will compile without error and will print the numbers 0 through 10 when run.
d. The code will compile without error and will print the number 0 when run.
e. The code will compile without error and will print the number 0 twice when run.
f. The code will compile without error and will print the numbers 1 through 10 when run.

4.29) Which of the following implementations of a max() method will correctly return the largest value?

```
// (1)
      int max(int x, int y) {
      return (if (x > y) { x; } else { y; });
}
// (2)
int max(int x, int y) {
      return (if (x > y) { return x; } else { return y; });
}
// (3)
int max(int x, int y) {
      switch (x < y) {
            case true:
            return y;
            default:
            return x;
      };
}
// (4)
int max(int x, int y) {
      if (x>y) return x;
      return y;
}
```

Select the one correct answer.

a. Implementation labeled (1).
b. Implementation labeled (2).
c. Implementation labeled (3).
d. Implementation labeled (4).

4.30) Given the following code, which statement is true?

```
class MyClass {
      public static void main(String[] args) {
            int k=0;
            int l=0;
            for (int i=0; i <= 3; i++) {
                  k++;
                  if (i == 2) break;
                  l++;
            }
            System.out.println(k + ", " + l);
      }
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will print 3, 3 when run.
c. The program will print 4, 3 when run if break is replaced by continue.
d. The program will fail to compile if break is replaced by return.
e. The program will fail to compile if break is simply removed.

4.31) Which statements are true? Select the two correct answers.

a. {{}} is a valid statement block.
b. { continue; } is a valid statement block.
c. block: { break block; } is a valid statement block.
d. block: { continue block; } is a valid statement block.
e. The break statement can only be used in a loop (while, do-while or for) or a switch statement.

4.32) Which digits, and in which order, will be printed when the following program is run?

```
public class MyClass {
      public static void main(String[] args) {
            int k=0;
            try {
```

```
                int i = 5/k;
            } catch (ArithmeticException e) {
                System.out.println("1");
            } catch (RuntimeException e) {
                System.out.println("2");
                return;
            } catch (Exception e) {
                System.out.println("3");
            } finally {
                System.out.println("4");
            }
            System.out.println("5");
        }
    }
```

Select the one correct answer.

a. The program will only print 5.
b. The program will only print 1 and 4, in that order.
c. The program will only print 1, 2, and 4, in that order.
d. The program will only print 1, 4, and 5, in that order.
e. The program will only print 1, 2, 4, and 5, in that order.
f. The program will only print 3 and 5, in that order.

4.33) Given the following program, which statements are true?

```
public class Exceptions {
    public static void main(String[] args) {
        try {
            if (args.length == 0) return;
            System.out.println(args[0]);
        } finally {
            System.out.println("The end");
        }
    }
}
```

Select the two correct answers.

a. If run with no arguments, the program will produce no output.
b. If run with no arguments, the program will print" The end".
c. The program will throw an ArrayIndexOutOfBoundsException.
d. If run with one argument, the program will simply print the given argument.
e. If run with one argument, the program will print the given argument followed by"T he end".

4.34) What will be the result of attempting to compile and run the following program?

```
public class MyClass {
    public static void main(String[] args) {
        RuntimeException re = null;
        throw re;
    }
}
```

Select the one correct answer.

a. The code will fail to compile, since the main() method does not declare that it throws RuntimeException in its declaration.
b. The program will fail to compile, since it cannot throw re.
c. The program will compile without error and will throwj ava.lang.RuntimeException when run.
d. The program will compile without error and will throwj ava.lang.NullpointerException when run.
e. The program will compile without error and will run and terminate without any output.

4.35) Which statements are true? Select the two correct answers.

a. If an exception is uncaught in a method, the method will terminate and normal execution will resume.
b. An overriding method must declare that it throws the same exception classes as the method it overrides.
c. The main() method of a program can declare that it throws checked exceptions.

d. A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class.
e. finally blocks are executed if, and only if, an exception gets thrown while inside the correspondingtr y block.

4.36) Which digits, and in which order, will be printed when the following program is compiled and run?

```
public class MyClass {
      public static void main(String[] args) {
            try {
                  f();
            } catch (InterruptedException e) {
                  System.out.println("1");
                  throw new RuntimeException();
            } catch (RuntimeException e) {
                  System.out.println("2");
                  return;
            } catch (Exception e) {
                  System.out.println("3");
            } finally {
                  System.out.println("4");
            }
                  System.out.println("5");
            }
            // InterruptedException is a direct subclass of Exception.
            static void f() throws InterruptedException {
                  throw new InterruptedException("Time for lunch.");
      }
}
```

Select the one correct answer.
a. The program will print 5.
b. The program will print 1 and 4, in that order.
c. The program will print 1, 2, and 4, in that order.
d. The program will print 1, 4, and 5, in that order.
e. The program will print 1, 2, 4, and 5, in that order.
f. The program will print 3 and 5, in that order.

4.37) Which digits, and in which order, will be printed when the following program is run?

```
public class MyClass {
public static void main(String[] args) throws InterruptedException {
try {
f();
System.out.println("1");
} finally {
System.out.println("2");
}
System.out.println("3");
}
// InterruptedException is a direct subclass of Exception.
static void f() throws InterruptedException {
throw new InterruptedException("Time to go home.");
}
}
```

Select the one correct answer.
a. The program will print 2 and throw InterruptedException.
b. The program will print 1 and 2, in that order.
c. The program will print 1, 2, and 3, in that order.
d. The program will print 2 and 3, in that order.
e. The program will print 3 and 2, in that order.
f. The program will print 1 and 3, in that order.

4.38) What is wrong with the following code?

```
public class MyClass {
```

```
public static void main(String[] args) throws A {
try {
f();
} finally {
System.out.println("Done.");
} catch (A e) {
throw e;
}
}
public static void f() throws B {
throw new B();
}
}
class A extends Throwable {}
class B extends A {}
```

Select the one correct answer.

a. The main() method must declare that it throwsB .
b. The finally block must follow the catch block in the main() method.
c. The catch block in the main() method must declare that it catchesB rather than A.
d. A single try block cannot be followed by both af inally and a catch block.
e. The declaration of class A is illegal.

4.39) What is the minimal list of exception classes that the overriding method f() in the following code must declare in its throws clause before the code will compile correctly?

```
class A {
// InterruptedException is a direct subclass of Exception.
void f() throws ArithmeticException, InterruptedException {
div(5, 5);
}
int div(int i, int j) throws ArithmeticException {
return i/j;
}
}
public class MyClass extends A {
void f() /* throws [...list of exceptions...] */ {
try {
div(5, 0);
} catch (ArithmeticException e) {
return;
}
throw new RuntimeException("ArithmeticException was expected.");
}
}
```

Select the one correct answer.

a. Does not need to specify any exceptions.
b. Needs to specify that it throws ArithmeticException.
c. Needs to specify that it throws InterruptedException.
d. Needs to specify that it throws RuntimeException.
e. Needs to specify that it throws bothA rithmeticException and InterruptedException.

4.40) What, if anything, would cause the following code not to compile?

```
class A {
void f() throws ArithmeticException {
//...
}
}
public class MyClass extends A {
public static void main(String[] args) {
A obj = new MyClass();
try {
obj.f();
} catch (ArithmeticException e) {
```

```
return;
} catch (Exception e) {
System.out.println(e);
throw new RuntimeException("Something wrong here");
}
}
// InterruptedException is a direct subclass of Exception.
void f() throws InterruptedException {
//...
}
}
```

Select the one correct answer.

a. The main() method must declare that it throwsR untimeException.
b. The overriding f() method in MyClass must declare that it throws ArithmeticException, since the f() method in class A declares that it does.
c. The overriding f() method in MyClass is not allowed to throw InterruptedException, since the f() method in class A does not throw this exception.
d. The compiler will complain that the catch(ArithmeticException) block shadows the catch(Exception) block.
e. You cannot throw exceptions from a catch block.
f. Nothing is wrong with the code, it will compile without errors.

4.41) Assuming assertions are enabled, which of these assertion statements will throw an error? Select the two correct answers.

a. assert true : true;
b. assert true : false;
c. assert false : true;
d. assert false : false;

4.42) Which of the following are valid runtime options? Select the two correct answers.

a. -ae
b. -enableassertions
c. -source 1.4
d. -disablesystemassertions
e. -dea

4.43) What is the class name of the exception thrown by an assertion statement? Select the one correct answer.

a. Depends on the assertion statement.
b. FailedAssertion
c. AssertionException
d. RuntimeException
e. AssertionError
f. Error

4.44) What can cause an assertion statement to be ignored? Select the one correct answer.

a. Nothing.
b. Using appropriate compiler options.
c. Using appropriate runtime options.
d. Using both appropriate compiler and runtime options.

4.45) Given the following method, which statements will throw an exception, assuming assertions are enabled?

```
static int inv(int value) {
assert value > -50 : value < 100;
return 100/value;
}
```

Select the two correct answers.

a. inv(-50);
b. inv(0);
c. inv(50);

d. inv(100);
e. inv(150);

4.46) Which runtime options would cause assertions to be enabled for the class org.example.ttp.Bottle? Select the two correct answers.

a. -ea
b. -ea:Bottle
c. -ea:org.example
d. -ea:org...
e. -enableexceptions:org.example.ttp.Bottle
f. -ea:org.example.ttp

4.47) What will be the result of compiling and running the following code with assertions enabled?

```
public class TernaryAssertion {
public static void assertBounds(int low, int high, int value) {
assert ( value > low ? value < high : false )
: (value < high ? "too low" : "too high" );
}
public static void main(String[] args) {
assertBounds(100, 200, 150);
}
}
```

Select the one correct answer.

a. The compilation fails because the method name assertBounds cannot begin with the keyword assert.
b. The compilation fails because the assert statement is invalid.
c. The compilation succeeds and the program runs without errors.
d. The compilation succeeds and an AssertionError with the error message "too low" is thrown.
e. The compilation succeeds and an AssertionError with the error message "too high" is thrown.

4.48) Which statements are true about the AssertionError class? Select the two correct answers.

a. It is a checked exception.
b. It has a method named toString().
c. It has a method named getErrorMessage().
d. It can be caught by at ry-catch construct.

4.49) Which of these classes is the direct superclass of AssertionError? Select the one correct answer.

a. Object
b. Throwable
c. Exception
d. Error
e. RuntimeError

4.50) Given the following command, which classes would have assertions enabled?

```
java -ea -da:com... net.example.LaunchTranslator
```

Select the two correct answers.

a. com.example.Translator
b. java.lang.String
c. dot.com.Boom
d. net.example.LaunchTranslator
e. java.lang.AssertionError

Flow Control, Exception Handling & Assertions Answers

4.1) (a), (d)
(b) is wrong because the return value in line 4 is used. (c) is wrong because a NullPointerException is thrown in line 3 and is not caught in the method, line 7 is never reached.

4.2) (b), (d)
(b) is correct because exceptions obey a hierarchy just like other objects. Because these exceptions descend from RuntimeException, they do not have to be declared. Therefore, answer (d) is correct. The significant word here is "must". Because these exceptions descend from RuntimeException, they do not have to be declared. Therefore, answer (a) is incorrect. Answer (c) is incorrect for a similar reason, because these exceptions descend from RuntimeException. They do not have to be caught even if declared by method *X*.

4.3) (a)
It will give you error for unreached statement.All other statements are valid.

4.4) (a)
You cannot use any checked exception in a catch block if it never thrown.

4.5) (b)
You can use return(without any value) in a method which is declared void.As return is used it will not print "C" but at the same time there will be no error for that.

4.6) (c)
As this time we have used finally so it will print "A" and "C".Finaly will always execute irrespective of whether exception is thrown or not.

4.7) (b)
System.exit() will restrict the finally block to be executed.

4.8) (a)
This will fail to compile because the labelled block does not enclose the break statement.

4.9) (a)
This will not compile because B is a subclass of A, so it must come before A. For multiple catch statements the rule is to place the the subclass exceptions before the superclass exceptions.

4.10) (a)
The boolean condition here evaluates to true so the print line will execute once.

4.11) (c)
This will not compile because the print line statement is unreachable.

4.12) (a)

4.13) (c)

4.14) (d)
Counter never gets incremented because the inner loop is never entered.

4.15) (b)

4.16) (b)

4.17) (a)

4.18) (d)

The program will display the letter b when run. The second if statement is evaluated since the boolean expression of the first if statement is true. The else clause belongs to the second if statement. Since the boolean expression of the second if statement is false, the if block is skipped and the else clause is executed.

4.19) (a), (b), and (e)
The conditional expression of an if statement can have any subexpressions, including method calls, as long as the whole expression evaluates to a value of type boolean. The expression (a = b) does not compare the variables a and b, but assigns the value of b to the variable a. The result of the expression is the value being assigned. Since a and b are boolean variables, the value returned by the expression is also boolean. This allows the expression to be used as the condition for an if statement. An if statement must always have an if block, but the else clause is optional. The expression if (false) ; else ; is legal. In this case, both the if block and the else block are simply the empty statement.

4.20) (f)
There is nothing wrong with the code. The case and default labels do not have to be specified in any specific order. The use of the break statement is not mandatory, and without it the control flow will simply fall through the labels of thes witch statement.

4.21) (a)
The type of the switch expression must be either byte, char, short, or int. This excludes (b) and (e). The type of the case labels must be assignable to the type of the switch expression. This excludes (c) and (d).

4.22) (e)
The loop body is executed twice and the program will print 3. The first time the loop is executed, the variable i changes from 1 to 2 and the variable b changes from false to true. Then the loop condition is evaluated. Since b is true, the loop body is executed again. This time the variable i changes from 2 to 3 and the variable b changes from true to false. The loop condition is now evaluated again. Since b is now false, the loop terminates and the current value of i is printed.

4.23) (b) and (e)
Both the first and the second number printed will be 10. Both the loop body and the increment expression will be executed exactly 10 times. Each execution of the loop body will be directly followed by an execution of the increment expression. Afterwards, the condition j<10 is evaluated to see whether the loop body should be executed again.

4.24) (c)
Only (c) contains a valid for loop. The initializer in a for statement can contain either declarations or a list of expression statements, but not both as attempted in (a). The loop condition must be of type boolean. (b) tries to use an assignment of an int value (notice the use of = rather than ==) as a loop condition and is, therefore, not valid. The loop condition in the for loop (d) tries to use the uninitialized variable i, and the for loop in (e) is simply syntactically invalid.

4.25) (f)
The code will compile without error, but will never terminate when run. All the sections in the for header are optional and can be omitted (but not the semicolons). An omitted loop condition is interpreted as being true. Thus, a for loop with an omitted loop condition will never terminate, unless a break statement is encountered in the loop body. The program will enter an infinite loop at (4).

4.26) (b), (d), and (e)
The loop condition in a while statement is not optional. It is not possible to break out of the if statement in (c). Notice that if the if statement had been placed within a labeled block, as witch statement, or a loop, the usage ofb reak would be valid.

4.27) (a) and (d)
"i=1, j=0" and "i=2, j=1" are part of the output. The variable i iterates through the values 0, 1, and 2 in the outer loop, while j toggles between the values 0 and 1 in the inner loop. If the values oif and j are equal, the printing of the values is skipped and the execution continues with the next iteration of the outer loop. The following can be deduced when the program is run: variables i and j are both 0 and the execution continues with the next iteration of the outer loop.i= "1, j=0" is printed and the next iteration of the inner loop starts.

Variables i and j are both 1 and the execution continues with the next iteration of the outer loop. "i=2, j=0" is printed and the next iteration of the inner loop starts. "i=2, j=1" is printed, j is incremented, j < 2 fails, and the inner loop ends. Variable i is incremented, i < 3 fails, and the outer loop ends.

4.28) (b)
The code will fail to compile, since the conditional expression of the if statement is not of type boolean. The conditional expression of an if statement must be of type boolean. The variable i is of type int. There is no conversion between boolean and other primitive types.

4.29) (d)
Implementation (4) will correctly return the largest value. The if statement does not return any value and, therefore, cannot be used as in implementations (1) and (2). Implementation (3) is invalid since neither the switch expression nor the case label values can be of type boolean.

4.30) (c)
As it stands, the program will compile correctly and will print "3, 2" when run. If the break statement is replaced with a continue statement, the loop will perform all four iterations and will print 4", 3". If the break statement is replaced with a return statement, the whole method will end when i equals 2, before anything is printed. If the break statement is simply removed, leaving the empty statement (;), the loop will complete all four iterations and will print " 4, 4".

4.31) (a) and (c)
The block construct {} is a compound statement. The compound statement can contain zero or more arbitrary statements. Thus, {{}} is a legal compound statement, containing one statement that is also a compound statement, containing no statement. The block { continue; } by itself is not valid, since the continue statement cannot be used outside the context of a loop. (c) is a valid example of breaking out of a labeled block. (d) is not valid for the same reasons (b) was not valid. The statement at (e) is not true, since the break statement can also be used to break out of labeled blocks, as illustrated by (c).

4.32) (d)
The program will only print 1, 4, and 5, in that order. The expression 5/k will throw an ArithmeticExecption, since k equals 0. Control is transferred to the first catch block, since it is the first block that can handle arithmetic exceptions. This exception handler simply prints 1. The exception has now been caught and normal execution can resume. Before leaving the try statement, the finally block is executed. This block prints 4. The last statement of the main() method prints 5.

4.33) (b) and (e)
If run with one argument, the program will print the given argument followed by "The end". The finally block will always be executed, no matter how control leaves the try block.

4.34) (d)
The program will compile without error, but will throw a NullPointerException when run. The throw statement can only throw Throwable objects. A NullPointerException will be thrown if the expression of the throw statement results in a null reference.

4.35) (c) and (d)
Normal execution will only resume if the exception is caught by the method. The uncaught exception will propagate up the runtime stack until some method handles it. An overriding method need only declare that it can throw a subset of the checked exceptions the overridden method can throw. The main() method can declare that it throws checked exceptions just like any other method. The finally block will always be executed, no matter how control leaves the try block.

4.36) (b)
The program will print 1 and 4, in that order. An InterruptedException is handled in the first catch block. Inside this block a new RuntimeException is thrown. This exception was not thrown inside the try block and will not be handled by the catch blocks, but will be sent to the caller of the main() method. Before this happens, the finally block is executed. The code printing 5 is never reached, since the runtimeException remains uncaught after the execution of the finally block.

4.37) (a)

The program will print 2 and throw an InterruptedException. An InterruptedException is thrown in the try block. There is no catch block to handle the exception, so it will be sent to the caller of the main() method, that is, to the default exception handler. Before this happens, the finally block is executed. The code printing 3 is never reached.

4.38) (b)
The only thing that is wrong with the code is the ordering of the catch and finally blocks. If present, the finally block must always appear last in a try-catch-finally construct.

4.39) (a)
Overriding methods can specify all, none, or a subset of the checked exceptions the overridden method declares in its throws clause. The InterruptedException is the only checked exception specified in the throws clause of the overridden method. The overriding method f() need not specify any checked exception from the throws clause of the overridden method.

4.40) (c)
The overriding f() method in MyClass is not permitted to throw the checked InterruptedException, since the f() method in class A does not throw this exception. To avoid compilation errors, either the overridingf( ) method must not throw an InterruptedException or the overridden f() method must declare that it can throw an InterruptedException.

4.41) (c) and (d)
Statements (c) and (d) will throw an AssertionError because the first expression is false. Statement (c) will report true as the error message, while (d) will report false as the error message.

4.42) (b) and (d)
-ea (enable assertions) is a valid runtime option, not -ae. -source 1.4 is a compile time option. -dsa (disable system assertions) is a valid runtime option, not -dea.

4.43) (e)
The class of exceptions thrown by assertion statements is always AssertionError.
4.44) (c)
Assertions can be enabled or disabled at runtime, but the assert statements are always compiled into bytecode.
4.45) (a) and (b)
Statement (a) will cause the assert statement to throw an AssertionError since (-50 > -50) evaluates to false. Statement (b) will cause the expression 100/value to throw an ArithmeticException since an integer division by zero is attempted.
4.46) (a) and (d)
Option (a) enables assertions for all non-system classes, while (d) enables assertions for all classes in the package org and its subpackages. Options (b), (c), and (f) try to enable assertions in specifically named classesB: ottle, org.example, and org.example.ttp. Option (e) is not a valid runtime option.
4.47) (c)
The assert statement correctly asserts that 150 is greater than 100 and less than 200.
4.48) (b) and (d)
The AssertionError class, like all other error classes, is not a checked exception, and need not be declared in ath rows clause. After an AssertionError is thrown, it is propagated exactly the same way as other exceptions, and can be caught by a try-catch construct.
4.49) (d)
The class Error is the direct superclass of AssertionError.

4.50) (c) and (d)
The command line enables assertions for all non-system classes, except for those in the com package or one of its subpackages. Assertions are not enabled for the system classes in (b) and (e).

Object Oriented Programming Mock Questions

5.1) Assume we have the following code in the file /abc/def/Q.java:

```
//File: /abc/def/Q.java:
package def;

public class Q {
   private int privateVar;
   int packageVar;
   protected int protectedVar;
   public int publicVar;
}

And this code is in /abc/Tester.java:
//File: /abc/Tester.java:
import def.Q;

public class Tester extends Q {
   Q sup = new Q();
   Sub sub = new Sub();

   public void someMethod() {
      // First, try to refer to sup's memebers.
      sup.privateVar   = 1;  // Line 1
      sup.packageVar   = 2;  // Line 2
      sup.protectedVar = 3;  // Line 3
      sup.publicVar    = 4;  // Line 4

      // Next, try to refer to this object's members
      // supplied by class Q.
      privateVar       = 5;  // Line 5
      packageVar       = 6;  // Line 6
      protectedVar     = 7;  // Line 7
      publicVar        = 8;  // Line 8

      // Next, let's try to access the members of
      // another instance of Tester.
      Tester t = new Tester();
      t.privateVar     = 9;  // Line 9
      t.packageVar     = 10; // Line 10
      t.protectedVar   = 11; // Line 11
      t.publicVar      = 12; // Line 12

      // Finally, try to refer to the members in a
      // subclass of Tester.
      sub.privateVar   = 13; // Line 13
      sub.packageVar   = 14; // Line 14
      sub.protectedVar = 15; // Line 15
      sub.publicVar    = 16; // Line 16
   }
}
```

And this code is in /abc/Sub.java:

```
//File: /abc/Sub.java:
public class Sub extends Tester {
}
```

Assume the directory, /abc, is in the compiler's CLASSPATH. When you try to compile Tester.java there will be several compiler errors. For each of the labeled lines above, decide whether or not a compiler error will be generated.

5.2) Given the following listing of the Widget class:

```
class Widget extends Thingee {                          //(1)
    static private int widgetCount = 0;                 //(2)
    public String wName;                                //(3)
    int wNumber;                                        //(4)

    static synchronized int addWidget() {               //(5)
        wName = "I am Widget # " + widgetCount;         //(6)
        return widgetCount;                             //(7)
    }                                                   //(8)

    public Widget() {                                   //(9)
        wNumber = addWidget();                          //(10)
    }                                                   //(11)
}                                                       //(12)
```

What happens when you try to compile the class and use multiple Widget objects in a program?

(a) The class compiles and each Widget will get a unique wNumber and wName reflecting the order in which the Widgets were created.
(b) The compiler objects to line 6.
(c) The class compiles, but a runtime error related to the access of the variable wName occurs in the addWidget method.

5.3) The following method definition is designed to parse and return an integer from an input string that is expected to look like "nnn,ParamName." In the event of a NumberFormatException, the method is to return –1.

```
public int getNum (String s) {                          //(1)
    try {                                               //(2)
        String tmp = S.substring(0, S.indexOf(',',));    //(3)
        return Integer.parseInt(tmp);                   //(4)
    } catch (NumberFormatException e) {                 //(5)
        System.out.println("Problem in " + tmp);        //(6)
    }                                                   //(7)
    return -1;                                          //(8)
}                                                       //(9)
```

What happens when you try to compile this code and execute the method with an input string that does contain a comma separating the number from the text data?

(a) A compiler error in line 6 prevents compilation.
(b) The method prints the error message to standard output and returns –1
(c) A NullPointerException is thrown in line 3.
(d) A StringIndexOutOfBoundsException is thrown in line 3.

5.4) Your chief Software designer has shown you a sketch of the new Computer parts system she is about to create. At the top of the hierarchy is a Class called Computer and under this are two child classes. One is called LinuxPC and one is called WindowsPC.
The main difference between the two is that one runs the Linux operating System and the other runs the Windows System (of course another difference is that one needs constant re-booting and the other runs reliably). Under the WindowsPC are two Sub classes one called Server and one Called Workstation. How might you appraise your designers work?

(a) Give the goahead for further design using the current scheme.
(b) Ask for a re-design of the hierarchy with changing the Operating System to a field rather than Class type.
(c) Ask for the option of WindowsPC to be removed as it will soon be obsolete.
(d) Change the hierarchy to remove the need for the superfluous Computer Class.

5.5) Given the following class definition which of the following can be legally placed after the comment line //Here ?

```
class Base{
    public Base(int i){}
}
public class MyOver extends Base {
    public static void main(String arg[]) {
        MyOver m = new MyOver(10);
```

```
    }
    MyOver(int i){
        super(i);
    }
    MyOver(String s, int i){
        this(i);
        //Here
    }
}
```

(a) MyOver m = new MyOver();
(b) super();
(c) this("Hello",10);
(d) Base b = new Base(10);

5.6) Suppose you have two classes defines as follows:

```
class ApBase extends Object implements Runnable
class ApDerived extends ApBase implements Observer
```

Given two variables created as follows:

```
ApBase aBase = new ApBase();
ApDerived aDer = new ApDerived();
```

Which of the following Java statements will compile and execute without error?

(a) Runnable rn = aDer;
(b) Runnable rn2 = (Runnable) aBase;
(c) Observer ob = aBase;
(d) Observer ob2 = (Observer) aBase;

5.7) Suppose we have an ApBase class declared as:

```
class ApBase extends Object implements Runnable
```

The following code fragment takes a reference to an ApBase object and assigns it to a variety of variables. What will happen when you try to compile and run this code?

```
ApBase aBase = new ApBase();
Runnable aR = aBase;
Object obj = aR;
ApBase x = (ApBase)obj;
```

(a) The compiler objects to line 2
(b) The compiler objects to line 3
(c) The code compiles but when run throws a ClassCastException in line 4
(d) The code compiles and runs without problem.

5.8) Suppose you have two classes defines as follows:

```
class ApBase extends Object implements Runnable
class ApDerived extends ApBase implements Observer
```

Given two variables created as follows:

```
ApBase aBase = new ApBase();
ApDerived aDer = new ApDerived();
```

Which of the following Java statements will compile and execute without error?

(a) Object obj = aBase; Runnable rn = obj;
(b) Object obj = aBase; Runnable rn = (Runnable)obj;
(c) Object obj = aBase; Observer ob = (Observer)aBase;
(d) Object obj = aDer; Observer ob2 = obj;

5.9) What will happen if you attempt to compile and run the following code?

```
class Base {}
class Sub extends Base {}
class Sub2 extends Base {}

public class CEx {
    public static void main(String argv[]){
        Base b = new Base();
        Sub  s = (Sub) b;
    }
}
```

(a) Compile and run without error.
(b) Compile time Exception.
(c) Runtime Exception.

5.10) You have these files in the same directory. What will happen when you attempt to compile and run Class1.java if you have not already compiled Base.java ?

```
//Base.java
package Base;
class Base {
    protected void amethod() {
        System.out.println("amethod");
    }
}

//Class1.java
package Class1;
public class Class1 extends Base {
    public static void main(String argv[]){
        Base b = new Base();
        b.amethod();
    }
}
```

(a) Compile Error: Methods in Base not found
(b) Compile Error: Unable to access protected method in base class
(c) Compilation followed by the output "amethod"
(d) Compile error: Superclass Class1.Base of class Class1.Class1 not found

5.11) You are taking over an aquarium simulation project. Your predecessor had created a generic Fish class that includes an oxygenConsumption method declared as follows:

```
public float oxygenConsumption(float temperature)
```

The aquarium simulation sums oxygen consumption for all fish in the tank with the following code fragment, where fishes is an array of Fish object references:

```
float total = 0;
for (int i = 0; i < fishes.length; i++) {
    total += fishes[i].oxygenConsumption(t);
}
```

you are writing a subclass for a particular fish species. Your task is to provide a method with species-specific metabolism data that will transparently fit into the simulation. Do you want to overload or override the oxygenConsumption method?

(a) overload.
(b) override.

5.12) The GenericFruit class declares the following method to return a float number of calories in the average serving size:

```
    public float aveCalories()
```

Your Apple class, which extends GenericFruit, overrides this method. In a DietSelection class that extends Object, you want to use the GenericFruit method on an Apple object. Select the correct way to finish the statement in the following code fragment so the GenericFruit version of aveCalories is called using the gf reference, or select option (d)

```
GenericFruit gf = new Apple();
float cal = // finish this statement using gf
```

(a) gf.aveCalories();
(b) ((GenericFruit)gf).aveCalories();
(c) gf.super.aveCalories();
(d) There is no way to call the GenericFruit method.

5.13) What will be the result of compiling and running the given program?
Select one correct answer.

```
1  public class Child extends Parent
2  {
3     public static int test(int i)
```

```
4     {
5        return 20;
6     }
7     public static void main(String[] args)
8     {
9        Parent c = new Child();
10       System.out.println(c.test(10));
11    }
12 }
13 class Parent
14 {
15    public static int test(int i)
16    {
17       return 5;
18    }
19 }
```

(a) Compile time as we can't overide static methods.
(b) Run time error as we can't overide static methods.
(c) Program compiles correctly and prints 5 when executed.
(d) Program compiles correctly and prints 20 when executed.

5.14) What will be the result of compiling and running the given program?
Select one correct answer.

```
1  class sample
2  {
3     sample(String s)
4     {
5        System.out.println("String");
6     }
7     sample(Object o)
8     {
9           System.out.println("Object");
10    }
11 }
12 class constructor
13 {
14    public static void main(String arg[])
15    {
16     sample s1=new sample(null);
17    }
18 }
```

(a) Compile time error as call to constructor at line no. 16 is ambigious.
(b) Run time error as call to constructor at line no. 16 is ambigious.
(c) Program compiles correctly and prints "object" when executed.
(d) Program compiles correctly and prints "string" when executed.

5.15) What will be the result of compiling and running the given program?
Select one correct answer.

```
1  class sample
2  {
3     sample(String s)
4     {
5        System.out.println("String");
6     }
7     sample(StringBuffer sb)
8     {
9           System.out.println("StringBuffer");
10    }
11 }
12 class constructor
13 {
14    public static void main(String arg[])
15    {
```

```
16      sample s1=new sample(null);
17    }
18 }
```

(a) Compile time error as call to constructor at line no. 16 is ambigious.
(b) Run time error as call to constructor at line no. 16 is ambigious.
(c) Program compiles correctly and prints "StringBuffer" when executed.
(d) Program compiles correctly and prints "string" when executed.

5.16) What will be the result of compiling and running the given program?
Select one correct answer.

```
1   class A
2   {
3      void callme()
4      {
5         System.out.println("A");
6      }
7      int r=10;
8   }
9   class B extends A
10  {
11     public void callme()
12     {
13        System.out.println("B");
14     }
15     int r=20;
16  }
17  class Q16
18  {
19      public static void main(String args[])
20      {
21         A a = new B();
22         a.callme();
23         System.out.println(a.r);
24      }
25  }
```

(a) Compile time error.
(b) Program compiles correctly and prints "A" and 10 when executed.
(c) Program compiles correctly and prints "B" and 20 when executed.
(d) Program compiles correctly and prints "B" and 10 when executed.

5.17) Whether the following code compile or not?
Select any two.

```
1  class Base {}
2  class Agg extends Base
3  {
4     public String getFields()
5     {
6        String name =  "Agg";
7        return name;
8     }
9  }
10 public class Inheritence
11 {
12    public static void main(String argv[])
13    {
14       Base a = new Agg();
15       System.out.println(a.getFields());
16    }
17 }
```

(a) It will compile.
(b) It will not compile.
(c) It will compile if we cast the variable a for the object of class Agg like ((Agg) a).getFields()

(d) We can cast the variable a for the object of class Agg, but it is not required.

5.18) What is the result of compiling and running this program?

```
class Mammal{
    void eat(Mammal m){
    System.out.println("Mammal eats food");
    }
 }
 class Cattle extends Mammal{
    void eat(Cattle c){
    System.out.println("Cattle eats hay");
    }
 }

 class Horse extends Cattle{
     void eat(Horse h){
     System.out.println("Horse eats hay");
     }
 }
 public class Test{
   public static void main(String[] args){
     Mammal h = new Horse();
     Cattle c = new Horse();
     c.eat(h);
   }
 }
```

(a) prints "Mammal eats food"
(b) prints "Cattle eats hay"
(c) prints "Horse eats hay"
(d) Class cast Exception at runtime.

5.19) Comsider the following class hierarchy.

```
1. interface A{
2. public void method1();
3.  }
4.  class One implements A{
5.  public void method1(){
6. System.out.println("hello");
7. }
8. }
9. class Two extends One{}
10. public class Test extends Two{
1(a) public static void main(String[] args)
1(b) {
1(c) A a;
1(d) Two t = new Two();
1(e) a = t;
16. a.method1();
17. }
18. }
```

What will be the outcome on attempting to compile and run this ?

(a) Compiles and runs printing out "hello".
(b) Compilation error at line 16.
(c) The compiler raises an objection to the assignment at line 15.
(d) Throws a NoSuchMethodException at runtime.

5.20) What will happen if you try to compile and run this ?

```
interface A{
public void innerMeth();
}
       public class Test {
```

```
A a;
int memVar = 1;
       void aMethod(){
              a = new A(){
              public void innerMeth(){
              System.out.println(memVar);
              } };
 }

        public static void main(String[] args){
         Test t = new Test();
          t.a.innerMeth();
         }
        }
```

(a) Compiler error.
(b) NoSuchMethodException at runtime.
(c) Compiles and runs printing 1
(d) Throws a NullPointerException at runtime.

5.21) What will happen if you try to compile and run this code.

```
class Rectangle{
  public int area(int length , int width) {
    return  length * width;
  }
 }

class Square extends Rectangle{
   public int area(long length , long width) {
   return  (int) Math.pow(length ,2);
  }
 }
class Test{
  public static void main(String args[]) {
   Square r = new Square();
   System.out.println(r.area(5 , 4));
  }
 }
```

(a) Will not compile.
(b) Will compile and run printing out 20
(c) Runtime error
(d) Will compile and run printing out 25

5.22) What will be the result of attempting to compile and run this.

```
class Base{}
class Derived extends Base{}
public class Test {
  public static void main(String[] args){
    Derived d = (Derived) new Base();
  }
 }
```

(a) Will not compile
(b) Compiles and runs without error.
(c) Runtime error

5.23) What will this program print out ?

```
class Base{
  int value = 0;
  Base(){
  addValue();
  }
```

```
    void addValue(){
    value += 10;
    }
   int getValue(){
     return value;
   }
 }
class Derived extends Base{
  Derived(){
   addValue();
  }
 void addValue(){
 value +=  20;
  }
}
public class Test {
  public static void main(String[] args){
      Base b = new Derived();
      System.out.println(b.getValue());
  }
}
```

(a) 10
(b) 20
(c) 30
(d) 40

5.24) Almost the same code as in the previous question. The only difference is the methods are static now. What will it print now?

```
class Base{
   static  int value = 0;
     Base(){
     addValue();
     }
 static void addValue(){
     value += 10;
     }
    int getValue(){
      return value;
    }
 }
class Derived extends Base{
  Derived(){
   addValue();
  }
static void addValue(){
 value +=  20;
  }
}
public class Test {
  public static void main(String[] args){
      Base b = new Derived();
      System.out.println(b.getValue());
  }
}
```

(a) 10
(b) 20
(c) 30
(d) 40

5.25) What is the result of attempting to compile and run this ?

```
interface ITest{
     public void setVal();
```

```
}
public class Test {
    private String a;
    void aMethod(){
    final String b;
    ITest it = new ITest() {
         public void setVal(){
         a = "Hello";
         b = " World";
         }};
    it.setVal();
    System.out.println(a + b);
    }
    public static void main(String[] args) {
    Test t = new Test();
    t.aMethod();
    }
}
```

(a) Code will not compile
(b) Run time error
(c) Will compile and run printing "Hello"
(d) Will compile and run without any output

5.26) What is the result of attempting to compile and run this ?

```
class Base{
     String s = "Base";
     String show(){
     return s;
     }
}
class Derived extends Base{
     String s = "Derived";
     }
public class Test {
     void print(Base b){
     System.out.println(b.show());
     }
     void print(Derived d){
     System.out.println(d.show());
     }
     public static void main(String[] args){
     Test t = new Test();
     Base b = new Derived();
     t.print(b);
     }
}
```

(a) Code will not compile
(b) Run time error
(c) Will compile and run printing "Derived"
(d) Will compile and run printing "Base"

5.27) What is the result of attempting to compile and run this ?

```
interface ITest{
   public void setVal();
  }
  public class Test {
    private String a;
    void aMethod(){
    final String b = " World";
    ITest it = new ITest() {
                public void setVal(){
                a = "Hello" + b;
```

```
                    }};
        it.setVal();
        System.out.println(a);
        }
        public static void main(String[] args) {
        Test t = new Test();
        t.aMethod();
        }
    }
```

(a) Code will not compile
(b) Run time error
(c) Will compile and run printing "Hello World"
(d) Will compile and run printing "Hello"

5.28) Which statements are true? Select the two correct answers.
a. In Java the extends clause is used to specify inheritance.
b. The subclass of a non-abstract class can be declared abstract.
c. All the members of the superclass are inherited by the subclass.
d. A final class can be abstract.
e. A class in which all the members are declared private, cannot be declared public.

5.29) Which statements are true? Select the two correct answers.
a. Inheritance defines a has-a relationship between a superclass and its subclasses.
b. Every Java object has a public method named equals.
c. Every Java object has a public method named length.
d. A class can extend any number of other classes.
e. A non-final class can be extended by any number of classes.

5.30) Which statements are true? Select the two correct answers.
a. A subclass must define all the methods from the superclass.
b. It is possible for a subclass to define a method with the same name and parameters as a method defined by the superclass.
c. It is possible for a subclass to define a field with the same name as a field defined by the superclass.
d. It is possible for two classes to be the superclass of each other.

5.31) Given the following classes and declarations, which statements are true?

```
// Classes
class Foo {
private int i;
public void f() { /* ... */ }
public void g() { /* ... */ }
}
class Bar extends Foo {
public int j;
public void g() { /* ... */ }
}
// Declarations:
// ...
Foo a = new Foo();
Bar b = new Bar();
// ...
```

Select the three correct answers.
a. The Bar class is a legal subclass of Foo.
b. The statement b.f(); is legal.
c. The statement a.j = 5; is legal.
d. The statement a.g(); is legal.
e. The statement b.i = 3; is legal.

5.32) Which statement is true? Select the one correct answer.
a. Private methods cannot be overridden in subclasses.

b. A subclass can override any method in a superclass.
c. An overriding method can declare that it throws more exceptions than the method it is overriding.
d. The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.
e. The overriding method can have a different return type than the overridden method.

5.33) Given classes A, B, and C, where B extends A, and C extends B, and where all classes implement the instance method void doIt(). How can the doIt() method in A be called from an instance method in C? Select the one correct answer.

a. doIt();
b. super.doIt();
c. super.super.doIt();
d. this.super.doIt();
e. A.this.doIt();
f. ((A) this).doIt();
g. It is not possible.

5.34) What would be the result of attempting to compile and run the following code?

```
// Filename: MyClass.java
public class MyClass {
public static void main(String[] args) {
C c = new C();
System.out.println(c.max(13, 29));
}
}
class A {
int max(int x, int y) { if (x>y) return x; else return y; }
}
class B extends A{
int max(int x, int y) { return super.max(y, x) - 10; }
}
class C extends B {
int max(int x, int y) { return super.max(x+10, y+10); }
}
```

Select the one correct answer.

a. The code will fail to compile because the max() method in B passes the arguments in the call super.max(y, x) in the wrong order.
b. The code will fail to compile because a call to am ax() method is ambiguous.
c. The code will compile without errors and will print1 3 when run.
d. The code will compile without errors and will print2 3 when run.
e. The code will compile without errors and will print2 9 when run.
f. The code will compile without errors and will print3 9 when run.

5.35) Given the following code, which is the simplest print statement that can be inserted into thep rint() method?

```
// Filename: MyClass.java
public class MyClass extends MySuperclass {
public static void main(String[] args) {
MyClass object = new MyClass();
object.print();
}
public void print() {
// INSERT CODE HERE THAT WILL PRINT
// THE "Hello, world!" STRING FROM THE Message
// CLASS.
}
}
class MySuperclass {
Message msg = new Message();
}
class Message {
// The message that should be printed:
```

```
String text = "Hello, world!";
}
```

Select the one correct answer.

a. System.out.println(text);
b. System.out.println(Message.text);
c. System.out.println(msg.text);
d. System.out.println(object.msg.text);
e. System.out.println(super.msg.text);
f. System.out.println(object.super.msg.text);

5.36) Given the following code, which of these constructors can be added to MySubclass without causing a compile-time error?

```
class MySuper {
int number;
MySuper(int i) { number = i; }
}
class MySub extends MySuper {
int count;
MySub(int cnt, int num) {
super(num);
count=cnt;
}
// INSERT ADDITIONAL CONSTRUCTOR HERE
}
```

Select the one correct answer.

a. MySub() {}
b. MySub(int cnt) { count = cnt; }
c. MySub(int cnt) { super(); count = cnt; }
d. MySub(int cnt) { count = cnt; super(cnt); }
e. MySub(int cnt) { this(cnt, cnt); }
f. MySub(int cnt) { super(cnt); this(cnt, 0); }

5.37) Which statement is true? Select the one correct answer.

a. A super() or this() call must always be provided explicitly as the first statement in the body of a constructor.
b. If both a subclass and its superclass do not have any declared constructors, the implicit default constructor of the subclass will call super() when run.
c. If neither super() nor this() is declared as the first statement in the body of a constructor, thent his() will implicitly be inserted as the first statement.
d. If super() is the first statement in the body of a constructor, then this() can be declared as the second statement.
e. Calling super() as the first statement in the body of a constructor of a subclass will always work, since all superclasses have a default constructor.

5.38) What will the following program print when run?

```
// Filename: MyClass.java
public class MyClass {
public static void main(String[] args) {
B b = new B("Test");
}
}
class A {
A() { this("1", "2"); }
A(String s, String t) { this(s + t); }
A(String s) { System.out.println(s); }
}
class B extends A {
B(String s) { System.out.println(s); }
B(String s, String t) { this(t + s + "3"); }
B() { super("4"); };
}
```

Select the one correct answer.

a. It will simply print Test.
b. It will print Test followed by Test.
c. It will print 123 followed by Test.
d. It will print 12 followed by Test.
e. It will print 4 followed by Test.

5.39) Which statements are true about interfaces? Select the two correct answers.
a. Interfaces allow multiple implementation inheritance.
b. Interfaces can be extended by any number of other interfaces.
c. Interfaces can extend any number of other interfaces.
d. Members of an interface are never static.
e. Members of an interface can always be declared static.

5.40) Which of these field declarations are legal within the body of an interface? Select the three correct answers.
a. public static int answer = 42;
b. int answer;
c. final static int answer = 42;
d. public int answer = 42;
e. private final static int answer = 42;

5.41) Which statements are true about interfaces? Select the two correct answers.
a. The keyword extends is used to specify that an interface inherits from another interface.
b. The keyword extends is used to specify that a class inherits from an interface.
c. The keyword implements is used to specify that an interface inherits from another interface.
d. The keyword implements is used to specify that a class inherits from an interface.
e. The keyword implements is used to specify that a class inherits from another class.

5.42) Which statement is true about the following code?

```
// Filename: MyClass.java
abstract class MyClass implements Interface1, Interface2 {
public void f() { }
public void g() { }
}
interface Interface1 {
int VAL_A = 1;
int VAL_B = 2;
void f();
void g();
}
interface Interface2 {
int VAL_B = 3;
int VAL_C = 4;
void g();
void h();
}
```

Select the one correct answer.
a. Interface1 and Interface2 do not match, therefore, MyClass cannot implement them both.
b. MyClass only implements Interface1. Implementation for void h() from Interface2 is missing.
c. The declarations of void g() in the two interfaces conflict, therefore, the code will not compile.
d. The declarations of int VAL_B in the two interfaces conflict, therefore, the code will not compile.
e. Nothing is wrong with the code, it will compile without errors.

5.43) Given the following code, which declaration can be inserted at the indicated line without causing a compilation error?

```
interface MyConstants {
int r = 42;
int s = 69;
// INSERT CODE HERE
}
```

Select the two correct answers.

a. final double circumference = 2*Math.PI*r;
b. int total = total + r + s;
c. int AREA = r*s;
d. public static MAIN = 15;
e. protected int CODE = 31337;

5.44) Given the following program, which statement is true?

```
// Filename: MyClass.java
public class MyClass {
public static void main(String[] args) {
A[] arrA;
B[] arrB;
arrA = new A[10];
arrB = new B[20];
arrA = arrB; // (1)
arrB = (B[]) arrA; // (2)
arrA = new A[10];
arrB = (B[]) arrA; // (3)
}
}
class A {}
class B extends A {}
```

Select the one correct answer.

a. The program will fail to compile because of the assignment at (1).
b. The program will throw a java.lang.ClassCastException in the assignment at (2) when run.
c. The program will throw a java.lang.ClassCastException in the assignment at (3) when run.
d. The program will compile and run without errors, even if the (B[]) cast in the statements at (2) and (3) is removed.
e. The program will compile and run without errors, but will not do so if the (B[]) cast in statements at (2) and (3) is removed.

5.45) Which is the first line that will cause compilation to fail in the following program?

```
// Filename: MyClass.java
class MyClass {
public static void main(String[] args) {
MyClass a;
MySubclass b;
a = new MyClass(); // (1)
b = new MySubclass(); // (2)
a = b; // (3)
b = a; // (4)
a = new MySubclass(); // (5)
b = new MyClass(); // (6)
}
}
class MySubclass extends MyClass {}
```

Select the one correct answer.

a. Line labeled (1).
b. Line labeled (2).
c. Line labeled (3).
d. Line labeled (4).
e. Line labeled (5).
f. Line labeled (6).

5.46) Given the following definitions and reference declarations, which one of the following assignments is legal?

```
// Definitions:
interface I1 {}
interface I2 {}
class C1 implements I1 {}
class C2 implements I2 {}
class C3 extends C1 implements I2 {}
```

```
// Reference declarations:
// ...
C1 obj1;
C2 obj2;
C3 obj3;
// ...
```

Select the one correct answer.

a. obj2 = obj1;
b. obj3 = obj1;
c. obj3 = obj2;
d. I1 a = obj2;
e. I1 b = obj3;
f. I2 c = obj1;

5.47) Given the following class definitions and the reference declarations, what can be said about the statement y = (Sub) x?

```
// Class definitions:
class Super {}
class Sub extends Super {}
// Reference declarations
// ...
Super x;
Sub y;
// ...
```

Select the one correct answer.

a. Illegal at compile time.
b. Legal at compile time, but might be illegal at runtime.
c. Definitely legal at runtime, but the (Sub) cast is not strictly needed.
d. Definitely legal at runtime, and the (Sub) cast is needed.

5.48) Given the following class definitions and declaration statements, which one of these assignments is legal at compile time?

```
// Definitions:
interface A {}
class B {}
class C extends B implements A {}
class D implements A {}
// Declaration statements:
// [...]
B b = new B();
C c = new C();
D d = new D();
// [...]
```

Select the one correct answer.

a. c = d;
b. d = c;
c. A a = d;
d. d = (D) c;
e. c = b;

5.49) Which letters will be printed when the following program is run?

```
// Filename: MyClass.java
public class MyClass {
public static void main(String[] args) {
B b = new C();
A a = b;
if (a instanceof A) System.out.println("A");
if (a instanceof B) System.out.println("B");
if (a instanceof C) System.out.println("C");
if (a instanceof D) System.out.println("D");
}
}
```

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
```

Select the three correct answers.

a. A will be printed.
b. B will be printed.
c. C will be printed.
d. D will be printed.

5.50) Given three classes A, B, and C, where B is a subclass of A and C is a subclass of B, which one of these boolean expressions is true when an object denoted by reference o has actually been instantiated from class B as opposed to from A or C? Select the one correct answer.

a. (o instanceof B) && (!(o instanceof A))
b. (o instanceof B) && (!(o instanceof C))
c. !((o instanceof A) || (o instanceof B))
d. (o instanceof B)
e. (o instanceof B) && !((o instanceof A) || (o instanceof C))

5.51) When the following program is run, it will print all the letters I, J, C, and D. Is this statement true or false?

```
public class MyClass {
public static void main(String[] args) {
I x = new D();
if (x instanceof I) System.out.println("I");
if (x instanceof J) System.out.println("J");
if (x instanceof C) System.out.println("C");
if (x instanceof D) System.out.println("D");
}
}
interface I{}
interface J{}
class C implements I {}
class D extends C implements J {}
```

Select the one correct answer.

a. True.
b. False.

5.52) What will be the result of attempting to compile and run the following program?

```
public class Polymorphism {
public static void main(String[] args) {
A ref1 = new C();
B ref2 = (B) ref1;
System.out.println(ref2.f());
}
}
class A { int f() { return 0; } }
class B extends A { int f() { return 1; } }
class C extends B { int f() { return 2; } }
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will compile without error, but will throw aC lassCastException when run.
c. The program will compile without error and print0 when run.
d. The program will compile without error and print1 when run.
e. The program will compile without error and print2 when run.

5.53) What will be the result of attempting to compile and run the following program?

```
public class Polymorphism2 {
public static void main(String[] args) {
A ref1 = new C();
B ref2 = (B) ref1;
System.out.println(ref2.g());
```

```
}
}
class A {
private int f() { return 0; }
public int g() { return 3; }
}
class B extends A {
private int f() { return 1; }
public int g() { return f(); }
}

class C extends B {
public int f() { return 2; }
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will compile without error and print0 when run.
c. The program will compile without error and print1 when run.
d. The program will compile without error and print2 when run.
e. The program will compile without error and print3 when run.

5.54) Given the following code, which statements are true?

```
public interface HeavenlyBody { String describe(); }
class Star {
String starName;
public String describe() { return "star " + starName; }
}
class Planet extends Star {
String name;
public String describe() {
return "planet " + name + " orbiting star " + starName;
}
}
```

Select the two correct answers:

a. The code will fail to compile.
b. The use of inheritance is justified, since Planet is-a Star.
c. The code will fail to compile if the name starName is replaced with the name bodyName throughout the declaration of the Star class.
d. The code will fail to compile if the name starName is replaced with the name name throughout the declaration of the Star class.
e. An instance of Planet is a valid instance of HeavenlyBody.

5.55) Given the following code, which statement is true?

```
public interface HeavenlyBody { String describe(); }
class Star implements HeavenlyBody {
String starName;
public String describe() { return "star " + starName; }
}
class Planet {
String name;
Star orbiting;
public String describe() {
return "planet " + name + " orbiting " + orbiting.describe();
}
}
```

Select the one correct answer:

a. The code will fail to compile.
b. The use of aggregation is justified, since Planet has-a Star.
c. The code will fail to compile if the name starName is replaced with the name bodyName throughout the declaration of the Star class.
d. The code will fail to compile if the name starName is replaced with the name name throughout the declaration of the Star class.

e. An instance of Planet is a valid instance of a HeavenlyBody.

Object Oriented Programming Answers

5.1) When one class extends another class, instances of the subclass will contain their own copies of **ALL** members declared in the superclass (including any private variables). In the example above all instances of Tester will contain all four of the variables declared in Q. However, not all of these variables will be accessible (i.e., visible) by instances of Tester.
In the descriptions below, we say a variable is *inherited* by a subclass if it is accessible to instances of the subclass.

sup.privateVar = 1; // Line 1
private members are only accessible from within the class in which they are declared. Because privateVar was declared in class Q, it is not accessible from class Tester. The following compiler error results:

```
Tester.java:10: privateVar has private access in def.Q
      sup.privateVar = 1;    // Line 1
        ^
```

sup.packageVar = 2; // Line 2
package members are only accessible to classes within the same package as the class that declares them. In the case, the member packageVar, is declared in class Q, in the def package. package is accessible only from within classes in the def package. Since the Tester class is in the "*default package*" it does not have access to packageVar. The following compiler error results:

```
Tester.java:11: packageVar is not public in def.Q; cannot be accessed from outside package
      sup.packageVar = 2;    // Line 2
        ^
```

sup.protectedVar = 3; // Line 3
Even though class Tester extends class Q, instances of Tester are not allowed access to protected members in instances of Q, proper. Because sup is an instance of Q, proper, a Tester instance will not have access to its protected members. An instance of class Tester can only access protected members in instances of classes that are of type Tester. This includes classes that inherit from Tester (such as the Sub class). The following compiler error results:

```
Tester.java:12: protectedVar has protected access in def.Q
      sup.protectedVar = 3;  // Line 3
        ^
```

sup.publicVar = 4; // Line 4
public members are available "*everywhere*". Consequently, Tester's reference to sup.publicVar compiles without error.
privateVar = 5; // Line 5
private members are only accessible from within the class in which they are declared. Because privateVar was declared in class Q, it is not accessible from class Tester. Note that on Line 1 we were trying to refer to a private member in another object (the member sup). On line 5 we are trying to refer the copy of privateVar that Tester contains by virtue of being a subclass of Q. In terms of the visibility of privateVar, however, it does not matter. privateVar is a private member declared in the Q class and can **ONLY** be accessed from within that class. The following compiler error results:

```
Tester.java:17: privateVar has private access in def.Q
      privateVar = 5;        // Line 5
      ^
```

packageVar = 6; // Line 6
As was true for line 2, package members are only accessible to classes within the same package as the class that declares them. Here we are attempting to access the packageVar member contained in an instance of the Tester class. In other words, an instance of Tester is trying to access the packageVar it contains by virture of its inheritance from Q. The protectedVar member was *declared* by a class in the def package and Tester is not in that package. Therefore, Tester cannot access the packageVar member. The following compiler error results:

```
Tester.java:18: packageVar is not public in def.Q; cannot be accessed from outside package
      packageVar = 6;        // Line 6
      ^
```

protectedVar = 7; // Line 7
There is no compiler error for line 7.

An instance of Tester can refer to its own copy of protectedVar, the copy of protectedVar contained in some other instance of Tester, or the copy of protectedVar contained in an instance of a subclass of Tester. An instance of Tester cannot, however, refer to a copy of protectedVar in an object that is *not* an instance of Tester.

publicVar = 8; // Line 8

As was the case with line 12, public members are available "*everywhere*". Consequently, Tester's reference to publicVar compiles without error.

t.privateVar = 9; // Line 9

As was the case with lines 1 and 5, the member, privateVar, is declared in class Q and can therefore only be accessed by instances of class Q, proper (i.e., non-subclasses of class Q). The following compiler error results:

```
Tester.java:25: privateVar has private access in def.Q
      t.privateVar = 9;      // Line 9
        ^
```

t.packageVar = 10; // Line 10

As was the case with lines 2 and 6, the member, packageVar, is declared in class Q which is in the def package. Only instances of classes in the same package as class Q can access the package variable, packageVar. Remember that because the Tester class does not have a package statement it is placed in Java's "*default package*". The following compiler error results:

```
Tester.java:26: packageVar is not public in def.Q; cannot be accessed from outside package
      t.packageVar = 10;     // Line 10
        ^
```

t.protectedVar = 11; // Line 11

You may want to refer back to the description of line 7. If we have a class (call it *B*) that contains a protected member (call it *p*) and another class (call it *S*) extends *B*, all instances of *S* can access protected members in any instance of class S, including subsclasses of *S*.

Once again, in this coded example, class *B* is class Q, class *S* is class Tester, and *p* is the protected member protectedVar. Line 11 is an example of an instance of Tester attempting to reference a protected member in another instance of Tester (represented by the member, t). Since an instance of Tester is attempting to reference a protected member in another instance of Tester, the line compiles cleanly.

t.publicVar = 12; // Line 12

As was the case with lines 4 and 8, public members are available "everywhere". Consequently, Tester's reference to t.publicVar compiles without error.

sub.privateVar = 13; // Line 13

As was the case with lines 1, 5, and 9, the member, sub.privateVar, is declared in class Q and can therefore only be accessed by instances of class Q, proper (i.e., non-subclasses of class Q). The following compiler error results:

```
Tester.java:32: privateVar has private access in def.Q
      sub.privateVar = 13;   // Line 13
        ^
```

sub.packageVar = 14; // Line 14

As was the case with lines 2, 6, and 10, the member, sub.packageVar, is declared in class Q which is in the def package. To access this member the class attempting the access (Tester, in this case) must be in the def package. Remember that since Tester does not have a package statement it is placed in Java's "default package". The following compiler error results:

```
Tester.java:33: packageVar is not public in def.Q; cannot be accessed from outside package
      sub.packageVar = 14;   // Line 14
        ^
```

sub.protectedVar = 15; // Line 15

You may want to refer back to the descriptions of lines 7 and 11. Actually, this line compiles cleanly for exactly the same reason line 11 compiled cleanly. Remember that class Sub extends class Tester. Consequently, any instances of Sub are also considered instances of Tester.

sub.publicVar = 16; // Line 16

As was the case with lines 4, 8 and 12, public members are available "everywhere". Consequently, Tester's reference to sub.publicVar compiles without error.

5.2) (b)

The static method addWidget cannot access the member variable wName. Static methods can refer to static variables only, such as widgetCount.

5.3) (a)
Because the scope of the tmp String is confined to the try block, thus, it cannot be used in line 6. answer (c) would not occur even if the scope of the tmp variable were fixed. answer (d) would occur only if the scope of the tmp variable were fixed by declaring and initializing tmp in the first line of the method.

5.4) (b)
This question is about the requirement to understand the difference between the "is-a" and the "has-a" relationship. Where a class is inherited you have to ask if it represents the "is-a" relationship. As the difference between the root and the two children are the operating system you need to ask are Linux and Windows types of computers.The answer is no, they are both types of Operating Systems. So option two represents the best of the options. You might consider having operating system as an interface instead but that is another story.
Of course there are as many ways to design an object hierarchy as ways to pronounce Bjarne Strousjoup, but this is the sort of answer that Sun will probably be looking for in the exam. Questions have been asked in discussion forums if this type of question really comes up in the exam. I think this is because some other mock exams do not contain any questions like this. I assure you that this type of question can come up in the exam. These types of question are testing your understanding of the difference between the is-a and has-a relationship in class design.

5.5) (d)
Any call to this or super must be the first line in a constructor. As the method already has a call to this, no more can be inserted.

5.6) (a), (b)
Answer (a) is correct because the ApDerived class inherits from ApBase, which implements Runnable. Answer (b) is also correct because the inserted cast (Runnable) is not needed but does not cause a problem. Answer (c) fails to compile because the compiler can tell that the ApBase class does not implement Observer. Answer (d) compiles, but fails to execute. Because of the specific cast the compiler thinks you know what you are doing, but the type of the aBase reference is checked when the statement executes and a ClassCastException is thrown.

5.7) (d)
These casts and assignment are all legal. (a) is incorrect incorrect because an object reference can be assigned to an interface reference as long as the compiler knows that the object implements the interface. Answer (b) is incorrect because an interface reference can be assigned to a reference to Object because Object is the base of the Java class hierarchy. Answer (c) is incorrect because the object referred to has not lost its identity, so it passes the runtime cast check.

5.8) (b)
It compiles and runs, the compiler assumes you know what you are doing with the cast to Runnable. Answer (a) fails to compile. As far as the compiler is concerned, obj is a plain Object so it objects to the assignment to a Runnable reference. Answer (c) compiles but fails to run. Because of the specific cast, the compiler thinks you know what you are doing, but the type of the aBase reference is checked when the statement executes, and a ClassCastException is thrown. Answer (d) fails to compile. As far as the compiler is concerned, obj is a plain Object so it objects to the assignment to an Observer reference.

5.9) (c)
Without the cast to sub you would get a compile time error. The cast tells the compiler that you really mean to do this and the actual type of b does not get resolved until runtime. Casting down the object hierarchy as the compiler cannot be sure what has been implemented in descendent classes. Casting up is not a problem because sub classes will have the features of the base classes. This can feel counter intuitive if you are aware that with primitives casting is allowed for widening operations (ie byte to int).

5.10) (d)
Using the package statement has an effect similar to placing a source file into a different directory. Because the files are in different packages they cannot see each other. The stuff about File1 not having been compiled was just to mislead, java has the equivalent of an "automake", whereby if it was not for the package statements the other file would have been automatically compiled.

5.11) (b)
by overriding the oxygenConsumption method, the Java runtime will call the overriding method for all fish where a specific method is provided or the generic method if there is none. Answer (a) is incorrect because if you overloaded the oxygenConsumption method using a different method signature, the Java runtime would not call the specific method for Fish where a specific method was provided. It would always call the generic method.

5.12) (d)
There is no way for a class outside the GenericFruit hierarchy to call the GenericFruit method using an Apple reference. Answer (a) is incorrect because the runtime resolution of method calls finds the Apple method. Answer (b) is incorrect because this extra cast does not change the object type. Answer (c) does not create a valid Java statement.

5.13) (c)
As both the method in class Child and Parent are static the method call is bound at compile time with the class whose reference variable is taken i.e. Parent.

5.14) (d)
Whenever a method/Constructor has an argument which matches two different methods/Constructors definations then it will always call the most specific one. As in our case Object is a general class and is super class of all other classes so it will call the String version of the Constructor.

5.15) (a)
As in this case both classes are peer i.e. both the classes is at the same level in the hierarchy chart.

5.16) (d)
Method are bound at run time where as variables are bound at compile time.It is the type of object which will tell that which method is to be called(i.e. B in this case), but it is the type of reference which will tell you which variable is to be called(i.e. A in this case).

5.17) (b), (c)
The program will not compile as the there is no method named getFields() defined in the class Base.It will compile if we cast the reference variable a for the object of class Agg like ((Agg) a).getFields()

5.18) (a)
The method that will be called is the one from class Mammal. The reasons are quite obvious.

5.19) (a)
Object reference conversion is possible here. The old type which is class can be assigned to an interface type as long as the class implements that interface.

5.20) (d)
You will get a NullPointerException because the inner class object gets assigned to the reference a only after the aMethod() runs. You can prevent the exception by calling t.aMethod() before the inner anonymous class method is called.

5.21) (a)
This code will fail to compile because the compiler cannot resolve the method call here.

5.22) (c)

5.23) (d)

5.24) (c)

5.25) (a)

5.26) (d)

5.27) (c)

5.28) (a) and (b)
The extends clause is used to specify that a class extends another class. A subclass can be declared abstract regardless of whether the superclass was declared abstract. Private, overridden, and hidden members from the superclass are not inherited by the subclass. A class cannot be declared both abstract and final, since an abstract class needs to be extended to be useful and a final class cannot be extended. The accessibility of the class is not limited by the accessibility of its members. A class with all the members declared private can still be declared public.

5.29) (b) and (e)
Inheritance defines an is-a relation. Aggregation defines a has-a relation. The Object class has a public method named equals, but it does not have any method named length. Since all classes are subclasses of the Object class, they all inherit the equals() method. Thus, all Java objects have a public method named equals. In Java, a class can only extend a single superclass, but there is no limit on how many classes can extend a superclass.

5.30) (b) and (c)
A subclass need not redefine all the methods defined in the superclass. It is possible for a subclass to define a method with the same name and parameters as a method defined by the superclass, but then the return type should also be the same. This is called method overriding. A subclass can define a field that can hide a field defined in a superclass. Two classes cannot be the superclass of each other.

5.31) (a), (b), and (d)
Bar is a legal subclass of Foo that overrides the method g(). The statement a.j = 5 is not legal, since the member j in class Bar cannot be accessed through a Foo reference. The statement b.i = 3 is not legal either, since the private member i cannot be accessed from outside of the class Foo.

5.32) (a)
A method can be overridden by defining a method with the same signature (i.e., name and parameter list) and return type as the method in a superclass. Only instance methods that are accessible by their simple name can be overridden. A private method, therefore, cannot be overridden in subclasses, but the subclasses are allowed to define a new method with exactly the same signature. A final method cannot be overridden. An overriding method cannot exhibit behavior that contradicts the declaration of the original method. An overriding method, therefore, cannot return a different type of value or declare that it throws more exceptions than the original method in the superclass.

5.33) (g)
It is not possible to invoke the doIt() method in A from an instance method in class C. The method in C needs to call a method in a superclass two levels up in the inheritance hierarchy. The super.super.doIt() strategy will not work, since super is a keyword and cannot be used as an ordinary reference, nor accessed like a field. If the member to be accessed had been a field, the solution would be to cast the this reference to the class of the field and use the resulting reference to access the field. Field access is determined by the declared type of the reference, whereas the instance method to execute is determined by the actual type of the object denoted by the reference.

5.34) (e)
The code will compile without errors. None of the calls to a max() method are ambiguous. When the program is run, the main() method will call the max() method in C with the parameters 13 and 29. This method will call the max() method in B with the parameters 23 and 39. The max() method in B will in turn call the max() method in A with the parameters 39 and 23. The max() method in A will return 39 to the max() method in B. The max() method in B will return 29 to the max() method in C. The max() method in C will return 29 to the main() method.

5.35) (c)
The simplest way to print the message in the class Message would be to use System. out.println(msg.text). The main() method creates an instance of MyClass, which results in the creation of aM essage instance. The field msg denotes this Message object in MySuperclass and is inherited by the MyClass object. Thus, the message in the Message object can be accessed directly by msg.text in the print() method of MyClass.

5.36) (e)

The class MySuper does not have a default constructor. This means that constructors in subclasses must explicitly call the superclass constructor to provide the required parameters. The supplied constructor accomplishes this by calling super(num) in its first statement. Additional constructors can accomplish this either by calling the superclass constructor directly using the super() call, or by calling another constructor in the same class using theth is() call, which in turn calls the superclass constructor. (a) and (b) are not valid since they do not call the superclass constructor explicitly. (d) fails since the super() call must always be the first statement in the constructor body. (f) fails since the super() and this() calls cannot be combined.

5.37) (b)
In a subclass without any declared constructors, the implicit default constructor will call super(). The use of the super() and this() statements are not mandatory as long as the superclass has a default constructor. If neithesru per() nor this() is declared as the first statement in the body of a constructor, then super() will implicitly be the first statement. A constructor body cannot have both a super() and a this() statement. Calling super() will not always work, since a superclass might not have a default constructor.

5.38) (d)
The program will print 12 followed by Test. When the main() method is executed, it will create a new instance of B by giving "Test" as argument. This results in a call to the constructor of classB that has one String parameter. The constructor does not explicitly call any superclass constructor, but instead the default constructor of the superclass A is called implicitly. The default constructor of class A calls the constructor in A that has two String parameters, giving it the arguments "1", "2". This constructor calls the constructor with oneS tring parameter, passing the argument "12". This constructor prints the argument. Now the execution of all the constructors in A is completed, and execution continues in the constructor of B. This constructor now prints the original argument "Test" and returns to the main() method.

5.39) (b) and (c)
Interfaces do not have any implementations and only permit multiple interface inheritance. An interface can extend any number of other interfaces and can be extended by any number of interfaces. Fields in interfaces are always static and method prototypes in interfaces are never static.

5.40) (a), (c), and (d)
Fields in interfaces declare named constants, and are always public, static, and final. None of these modifiers are mandatory in a constant declaration. All named constants must be explicitly initialized in the declaration.

5.41) (a) and (d)
The keyword implements is used when a class inherits from an interface. The keyword extends is used when an interface inherits from another interface or a class inherits from another class.

5.42) (e)
The code will compile without errors. The class MyClass declares that it implements the interfaces Interface1 and Interface2. Since the class is declared abstract, it does not need to supply implementations for all the method prototypes defined in these interfaces. Any non-abstract subclasses of MyClass must provide the missing method implementations. The two interfaces share a common method prototype void g(). MyClass provides an implementation for this prototype that satisfies both Interface1 and Interface2. Both interfaces provide declarations of constants named VAL_B. This can lead to an ambiguity when referring to VAL_B by its simple name from MyClass. The ambiguity can be resloved by using fully qualified names: Interface1.VAL_B and Interface2.VAL_B. However, there are no problems with the code as it stands.

5.43) (a) and (c)
Declaration (b) fails since it contains an illegal forward reference to its own named constant. The field type is missing in declaration (d). Declaration (e) tries illegally to use the protected modifier, even though named constants always have public accessibility.

5.44) (c)
The program will throw a java.lang.ClassCastException in the assignment at (3) when run. The statement at (1) will compile since the assignment is done from a subclass reference to a superclass reference. The cast at (2) assures the compiler that arrA will refer to an object that can be referenced by arrB. This will work when run since arrA will refer to an object of type B[]. The cast at (3) will also assure the compiler that arrA will refer to an object that can be referenced by arrB. This will not work when run since arrA will refer to an object of type A[].

5.45) (d)
Line (4) will cause a compile-time error since it attempts to assign a reference value of a supertype object to a reference of a subtype. The type of the source reference value is MyClass and the type of the destination reference is MySubclass. Lines (1) and (2) will compile since the reference is assigned a reference value of the same type. Line (3) will also compile since the reference is assigned a reference value of a subtype.

5.46) (e)
Only the assignment I1 b = obj3 is valid. The assignment is allowed since C3 extends C1, which implements I1. Assignment obj2 = obj1 is not legal since C1 is not a subclass of C2. Assignments obj3 = obj1 and obj3 = obj2 are not legal since neither C1 nor C2 is a subclass of C3. Assignment I1 a = obj2 is not legal since C2 does not implement I1. Assignment I2 c = obj1 is not legal since C1 does not implement I2.

5.47) (b)
The statement would be legal at compile time, since the reference x might actually refer to an object of the type Sub. The cast tells the compiler to go ahead and allow the assignment. At runtime, the reference x may turn out to denote an object of the type Super instead. If this happens, the assignment will be aborted and aC lassCastException will be thrown.

5.48) (c)
Only A a = d is legal. The reference value in d can be assigned to a since D implements A. The statements c = d and d = c are illegal since there is no subtype-supertype relationship between classes C and D. Even though a cast is provided, the statement d = (D) c is illegal. The object referred to by c cannot possibly be of type D, since D is not a subclass of C. The statement c = b is illegal since assigning a reference value of a reference of type B to a reference of type C requires a cast.

5.49) (a), (b), and (c)
The program will print A, B, and C when run. The object denoted by reference a is of type C. The object is also an instance of A and B, since C is a subclass of B and B is a subclass of A. The object is not an instance of D.

5.50) (b)
The expression (o instanceof B) will return true if the object referred to by o is of type B or a subtype of B. The expression (!(o instanceof C)) will return true unless the object referred to by o is of type C or a subtype of C. Thus, the expression (o instanceof B) && (!(o instanceof C)) will only return true if the object is of type B or a subtype of B that is not C or a subtype of C. Given objects of classes A, B, and C, this expression will only return true for objects of class B.

5.51) (a)
The program will print all of I, J, C, and D when run. The object referred to by reference x is of class D. Class D extends class C and class C implements interface I. This makes I, J, and C supertypes of class D. A reference of type D can be cast to any of its supertypes and is, therefore, an instanceof these types.

5.52) (e)
The program will print 2 when System.out.println(ref2.f()) is executed. The object referenced by ref2 is of class C, but the reference is of type B. Since B contains a method f(), the method call will be allowed at compile time. During execution it is determined that the object is of class C, and dynamic method lookup will cause the overridden method in C to be executed.

5.53) (c)
The program will print 1 when run. The f() methods in A and B are private and are not accessible by the subclasses. Because of this, the subclasses cannot overload or override these methods, but simply define new methods with the same signature. The object being called is of class C. The reference used to access the object is of type B. Since B contains a method g(), the method call will be allowed at compile time. During execution it is determined that the object is of class C, and dynamic method lookup will cause the overridden method g() in B to be called. This method calls a method named f(). It can be determined during compilation that this can only refer to thef( ) method in B, since the method is private and cannot be overridden. This method returns the value 1, which is printed.

5.54) (c) and (d)

The code as it stands will compile without error. The use of inheritance in this code is not justifiable since, conceptually, a planet is-not-a star. The code will fail if the name of the field starName is changed in the Star class since the subclass Planet tries to access it using the name starName. An instance of Planet is not an instance of HeavenlyBody. Neither Planet nor Star implements HeavenlyBody.

5.55) (b)
The code will compile without error. The use of aggregation in this code is justifiable. The code will not fail to compile if the name of the field starName is changed in the Star class, since the Planet class does not try to access the field by name, but instead uses the public method describe() in the Star class to do that. An instance of Planet is not an instance of HeavenlyBody since it neither implements HeavenlyBody nor extends a class that implements HeavenlyBody.

Garbage Collection & Object Lifetime Mock Questions

6.1) Which of the following statements about Java grabage collection are true statements? [3]

(a) The following code will start the garabage collection mechanism:
```
Runtime.getRuntime().gc();
Thread.yield();
```
(b) The following code will start the garbage collection mechanism:
```
System.gc();
Thread.sleep(1000);
```
(c) The garbage collection Thread has a low priority.
(d) The method by which Java tracks unused objects is specified in the language definition.
(e) The method by which Java determines that a chunk of memory is garbage is up to the implementor of the JVM.

6.2) Which of the following statements about finalize methods is true? [3]

(a) The purpose of a finalize method is to recover system resources other than memory.
(b) The purpose of a finalize method is to recover memory and other system resources.
(c) You should always write a finalize method for every class.
(d) The order in which objects are created controls the order in which their finalize methods are called.

6.3) After which line the object initially refered by str is eligible for garbage collection? [8]

Select one correct answer.

```
1   class Garbage
2   {
3       public static void main(String arg[])
4       {
5            String str=new String("Hello");
6            String str1=str;
7            str=new String("Hi");
8            str1=new String("Hello Again");
9            return;
10      }
11  }
```

(a) After line no. 6
(b) After line no. 7
(c) After line no. 8
(d) After line no. 9

6.4) What is the result that will be printed out here ?

```
int i =  - 128;
System.out.println((byte)i << 1);
```

(a) 0
(b)-256
(c) 127
(d)-127
(e) -1

6.5) What is the result of the following operation?

```
int n = 8;
n = n & n + 1;
n<<=n/2;
```

(a) 32
(b) 12
(c) 8
(d) 24
(e)128
(f) 0

6.6) Observe the code below

```
public class Test {
int a[] = new int[4];
  void aMethod()
  {
  int b = 0 , index;
  a[a[b]] = a[b] =  b = 2;
  index = ??? ;
    System.out.println(a[index]);
  }
 }
```

What value assigned to the variable index will print a non zero result?

(a)0
(b)1
(c)2
(d)3

6.7) Will this test evaluate to true?

```
Object a = "hello";
String b = "hello";
if(a == b)
System.out.println("equal");
else
System.out.println("not equal");
```

(a)Yes
(b)No

6.8) True or False.

The garbage collector is required to makes sure that all objects held by soft references are garbage collected before the VM throws an OutOfMemoryError.

(a)True
(b)False

6.9) What will be the value of the variable x here

```
int x = 8 | 12  ^ 8;
```

(a) 4
(b) 8
(c)12
(d) 0

6.10) What will be the value of the variable x here

```
int x = 8 ^ 5  &  6;
```

(a) 0
(b) 4
(c) 6
(d) 8
(e)12

6.11) What is the result that will be printed out

```
public class Test {
 public static void main(String[] args)
  {
  int a =  -8;
  int b = ~ -33;
  a>>>=b;
  System.out.println(a);
  }
}
```

(a)1
(b)8
(c)-8
(d)0

(e)16

6.12) Is this legal ?

```
public class Test {
  public static void main(String[] args)
  {
  char c = 65;
  double d = 10;
  if(c==d)
  System.out.println("equal");
  else
  System.out.println("not equal");
  }
}
```

(a)No.
(b)Yes.

6.13) Select the valid assignments
(a) int i = (int)16.2d;
(b) byte b = (byte)(long)16.2;
(c) byte b = 128;
(d) float f = 16.2;
(e) byte b = (int)16.2;

6.14) Which statement is true? Select the one correct answer.
a. Objects can be explicitly destroyed using the keyword delete.
b. An object will be garbage collected immediately after it becomes unreachable.
c. If object obj1 is accessible from object obj2, and object obj2 is accessible from obj1, then obj1 and obj2 are not eligible for garbage collection.
d. Once an object has become eligible for garbage collection, it will remain eligible until it is destroyed.
e. If object obj1 can access object obj2 that is eligible for garbage collection, then obj1 is also eligible for garbage collection.

6.15) Identify the location in the following program where the object, initially referenced with arg1, is eligible for garbage collection.

```
public class MyClass {
public static void main(String[] args) {
String msg;
String pre = "This program was called with ";
String post = " as first argument.";
String arg1 = new String((args.length > 0) ?
"'" + args[ 0 ] + "'" :
"<no argument>");
msg = arg1;
arg1 = null; // (1)
msg = pre + msg + post; // (2)
pre = null; // (3)
System.out.println(msg);
msg = null; // (4)
post = null; // (5)
args = null; // (6)
}
}
```

Select the one correct answer.
a. After the line labeled (1).
b. After the line labeled (2).
c. After the line labeled (3).
d. After the line labeled (4).
e. After the line labeled (5).
f. After the line labeled (6).

6.16) Which statement is true? Select the one correct answer.

a. If an exception is thrown during the execution of the finalize() method of an eligible object, then the exception is ignored and the object is destroyed.
b. All objects have a finalize() method.
c. Objects can be destroyed by explicitly calling thef inalize() method.
d. The finalize() method can be declared with any accessibility.
e. The compiler will fail to compile code that defines an overriding finalize() method that does not explicitly call the overridden finalize() method from the superclass.

6.17) Which statement is true? Select the one correct answer.

a. The compiler will fail to compile code that explicitly tries to call thef inalize() method.
b. The finalize() method must be declared with protected accessibility.
c. An overriding finalize() method in any class can always throw checked exceptions.
d. The finalize() method can be overloaded.
e. The body of the finalize() method can only access other objects that are eligible for garbage collection.

6.18) Which statement describes guaranteed behavior of the garbage collection and finalization mechanisms? Select the one correct answer.

a. Objects will not be destroyed until they have no references to them.
b. The finalize() method will never be called more than once on an object.
c. An object eligible for garbage collection will eventually be destroyed by the garbage collector.
d. If object A became eligible for garbage collection before object B, then object A will be destroyed before object B.
e. An object, once eligible for garbage collection, can never become accessible by a live thread.

6.19) Given the following class, which of these static initializer blocks can be inserted after the comment?

```
public class MyClass {
private static int count = 5;
final static int STEP = 10;
boolean alive;
// INSERT STATIC INITIALIZER BLOCK HERE
}
```

Select the three correct answers.

a. static { alive = true; count = 0; }
b. static { STEP = count; }
c. static { count += STEP; }
d. static ;
e. static {;}
f. static { count = 1; }

6.20) What will be the result of attempting to compile and run the following code?

```
public class MyClass {
public static void main(String[] args) {
MyClass obj = new MyClass(1);
}
static int i = 5;
static int l;
int j = 7;
int k;
public MyClass(int m) {
System.out.println(i + ", " + j + ", " + k + ", " + l + ", " + m);
}
{ j = 70; l = 20; } // Instance Initializer Block
static { i = 50; } // Static Initializer Block
}
```

Select the one correct answer.

a. The code will fail to compile, since the instance initializer block tries to assign a value to a static field.
b. The code will fail to compile, since the field k will be uninitialized when it is used.

c. The code will compile without error and will print5 0, 70, 0, 20, 0 when run.
d. The code will compile without error and will print5 0, 70, 0, 20, 20 when run.
e. The code will compile without error and will print5 , 70, 0, 20, 0 when run.
f. The code will compile without error and will print5 , 7, 0, 20, 0 when run.

6.21) Given the following class, which instance initializer block inserted at the indicated location will allow the class to compile without errors?

```
public class MyClass {
static int gap = 10;
double length;
final boolean active;
// INSERT CODE HERE
}
```

Select the one correct answer.

a. instance { active = true; }
b. MyClass { gap += 5; }
c. { gap = 5; length = (active ? 100 : 200) + gap; }
d. { ; }
e. { length = 4.2; }
f. { active = (gap > 5); length = 5.5 + gap;}

6.22) What will be the result of attempting to compile and run the following program?

```
public class Initialization {
private static String msg(String msg) {
System.out.println(msg); return msg;
}
public Initialization() { m = msg("1"); }
{ m = msg("2"); }
String m = msg("3");
public static void main(String[] args) {
Object obj = new Initialization();
}
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will compile without error and will print 1, 2, and 3 when run.
c. The program will compile without error and will print 2, 3, and 1 when run.
d. The program will compile without error and will print3 , 1, and 2 when run.
e. The program will compile without error and will print1 , 3, and 2 when run.

6.23) What will be the result of attempting to compile and run the following program?

```
public class Initialization {
private static String msg(String msg) {
System.out.println(msg); return msg;
}
static String m = msg("1");
{ m = msg("2"); }
static { m = msg("3"); }
public static void main(String[] args) {
Object obj = new Initialization();
}
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will compile without error and will print1 , 2, and 3 when run.
c. The program will compile without error and will print2 , 3, and 1 when run.
d. The program will compile without error and will print3 , 1, and 2 when run.
e. The program will compile without error and will print1 , 3, and 2 when run.

6.24) Which of the labeled lines in the following code can be uncommented by removing the // characters and still allow the code to compile correctly?

```
class GeomInit {
// int width = 14; /* Line A */
{
// area = width * height; /* Line B */
}
int width = 37;
{
// height = 11; /* Line C */
}
int height, area;
// area = width * height; /* Line D */
{
// int width = 15; /* Line E */
area = 100;
}
};
```

Select the two correct answers.

a. Line A
b. Line B
c. Line C
d. Line D
e. Line E

Garbage Collection & Object Lifetime Answers

6.1) (c), (e)

Answer (c) is correct because the JVM ssigns a low priority to the garbage collection thread. Answer (e) is also correct because picking the best garbage collection method is left up to the individual JVM implementor. Answer (a), (b) are incorrect because of the phrase "*will start*" these code fragments "*may*" start the garbage collector, but there is no guarantee. Answer (d) is incorrect because the langauge specification does not prescribe a method.

6.2) (a)

Answer (b) is incorrect because memory is recovered by garbage collection; finalizers are for other resources. Answer (c) is incorrect because objects that do not use system resources other than memory do not need finalizers. Answer (d) is incorrect because there is no guarantee about the order of object finalization.

6.3) (c)

At line no. 7 str refers to a new object but its previous object is still refered by str1 at that time.At line no. 8 str1 changes its reference then the object initially refered by str is eligible for garbage collection.

6.4) (b)

Though the value in i is cast to a byte it reverts to an int as the result of the left shift which makes the value of i - 256

6.5) (e)

The arithmetic + operator takes precedence over the bitwise & operator so 8 & 9 = 8 Shifting 8 by four bits to the left gives you 128

6.6) (a)

the operands are evaluated from left to right here so in the expression a[a[b]] the value of b is 0 and a[0] also holds the value 0 when the expression is evaluated so the array element at index 0 is assigned the value 2.

6.7) (a)

It does not matter if the object reference is of type Object this will still return true because both of them point to the same String object in the string pool.

6.8) (a)

If the VM finds itself unable to allocate memory for a new object it is required to kick start the garbage collector and reclaim all objects that are not held by strong references before throwing an OutOfMemoryError.

6.9) (c)

6.10) (e)

The precedence of bitwise operators is AND , XOR and OR so 5 AND 6 is evaluated first which gives you 4. 8 XOR 4 = 12

6.11) (c)

6.12) (b)

6.13) (a), (b), (e)

6.14) (e)

An object is only eligible for garbage collection if all remaining references to the object are from other objects that are also eligible for garbage collection. Therefore, if an object obj2 is eligible for garbage collection and object obj1 contains a reference to it, then object obj1 must also be eligible for garbage collection. Java does not have a keyword delete. An object will not necessarily be garbage collected immediately after it becomes unreachable. However, the object will be eligible for garbage collection.

Circular references do not prevent objects from being garbage collected, only reachable references do. An object is not eligible for garbage collection as long as the object can be accessed by any live thread. An object that has been eligible for garbage collection can be made non-eligible. This occurs if the finalize() method of the object creates a reachable reference to the object.

6.15) (b)
Before (1), the String object initially referenced by arg1 is denoted by both msg and arg1. After (1), the String object is only denoted by msg. At (2), reference msg is assigned a new reference value. This reference value denotes a new String object created by concatenating several other String objects. After (2), there are no references to the String object initially referenced by arg1. The String object is now eligible for garbage collection.

6.16) (b)
The Object class defines a protected finalize() method. All classes inherit from Object, thus, all objects have a finalize() method. Classes can override the finalize() method and, as with all overriding, the new method must not reduce the accessibility. The finalize() method of an eligible object is called by the garbage collector to allow the object to do any cleaning up, before the object is destroyed. When the garbage collector calls the finalize() method, it will ignore any exceptions thrown by the finalize() method. In all other cases, normal exception handling occurs when an exception is thrown during the execution of the finalize() method, that is, exceptions are not simply ignored. Calling the finalize() method does not in itself destroy the object. Chaining of the finalize() methods is not enforced by the compiler, and it is not mandatory to call the overridden finalize() method.

6.17) (d)
The finalize() method is like any other method, it can be called explicitly if it is accessible. However, the intended purpose of the method is to be called by the garbage collector in order to clean up before an object is destroyed. Overloading the finalize() method name is allowed, but only the method with the original signature will be called by the garbage collector. The finalize() method in Object is protected. This means that overriding methods must be declared either protected or public. The finalize() method in Object can throw any Throwable object. Overriding methods can limit the range of throwables to unchecked exceptions. Further overridden definitions of this method in subclasses will not be able to throw checked exceptions.

6.18) (b)
The finalize() method will never be called more than once on an object, even if thefi nalize() method resurrects the object. An object can be eligible for garbage collection even if there are references denoting the object, as long as the objects owning these references are also eligible for garbage collection. There is no guarantee that the garbage collector will destroy an eligible object before the program terminates. The order in which the objects are destroyed is not guaranteed. The finalize() method can make an object that has been eligible for garbage collection, accessible again by a live thread.

6.19) (c), (e), and (f)
Static initializer blocks (a) and (b) are not legal since the fields alive and STEP are non-static and final, respectively. (d) is syntactically not a legal static initializer block.

6.20) (c)
The program will compile without error and will print 50, 70, 0, 20, 0 when run. All fields are given default values unless they are explicitly initialized. Field i is assigned the value 50 in the static initializer block that is executed when the class is initialized. This assignment will override the explicit initialization of field i in its declaration statement. When the main() method is executed, the static field i is 50 and the static field l is 0. When an instance of the class is created using the new operator, the value of static field l (i.e., 0) is passed to the constructor. Before the body of the constructor is executed, the instance initializer block is executed, which assigns values 70 and 20 to fields j and l, respectively. When the body of the constructor is executed, the fields i, j, k, and l, and the parameter m, have the values 50, 70, 0, 20, and 0, respectively.

6.21) (f)
This class has a blank final boolean variable active. This variable must be initialized when an instance is constructed or else the code will not compile. The keyword static is used to signify that a block is a static initializer block. No keyword is used to signify that a block is an instance initializer block. (a) and (b) are not instance initializers blocks, and (c), (d), and (e) fail to initialize the blank final variable active.

6.22) (c)
The program will compile without error and will print 2, 3, and 1 when run. When the object is created and initialized, the instance initializer block is executed first, printing 2. Then the instance initializer expression is executed, printing 3. Finally, the constructor body is executed, printing 1. The forward reference in the instance initializer block is legal, as the use of the field m is on the left-hand side of the assignment.

6.23) (e)
The program will compile without error and will print 1, 3, and 2 when run. First, the static initializers are executed when the class is initialized, printing 1 and 3. When the object is created and initialized, the instance initializer block is executed, printing 2.

6.24) (c) and (e)
Line A will cause illegal redefinition of the field width. Line B uses an illegal forward reference to the fields width and height. The assignment in line C is legal. Line D is not a legal initializer since it is neither a declaration nor a block. Line E declares a local variable inside an initializer block.

Nested, Inner Classes & Interfaces Mock Questions

7.1) What will happen when you attempt to compile and run this program?

```
public class Outer {
    public String name = "Outer";
    public static void main(String argv[]) {
        Inner i = new Inner();
        i.showName();
    }//End of main

    private class Inner {
        String name =new String("Inner");
        void showName() {
            System.out.println(name);
        }
    }//End of Inner class
}
```

(a) Compile and run with output of "Outer".
(b) Compile and run with output of "Inner".
(c) Compile time error because Inner is declared as private.
(d) Compile time error because of the line creating the instance of Inner.

7.2) Which of the following statements are true?
(a) An inner class may be defined as static.
(b) There are NO circumstances where an inner class may be defined as private.
(c) An anonymous class may have only one constructor.
(d) An inner class may extend another class.

7.3) Given the following class definition, which of the following statements would be legal after the comment //Here?

```
class InOut{
    String s= new String("Between");
    public void amethod(final int iArgs) {
        int iam;
        class Bicycle{
            public void sayHello(){
                //Here
            }//End of bicycle class
        }
    }//End of amethod
    public void another(){
        int iOther;
    }
}
```

(a) System.out.println(s);
(b) System.out.println(iOther);
(c) System.out.println(iam);
(d) System.out.println(iArgs);

7.4) Which of the following statements are true?
(a) Adding more classes via import statements will cause a performance overhead, only import classes you actually use.
(b) Under no circumstances can a class be defined with the private modifier.
(c) A inner class may under some circumstances be defined with the protected modifier.
(d) An interface cannot be instantiated.

7.5) The following is an outline of code for top-level class. Assume that both classes have correct constructors taking no parameters.

```
class NormalClass {
    static class NestedClass {
        // methods and variables of NestedClass
```

```
    }
    // methods and variables of NestedClass
}
```

Which of the following code fragments shows the correct way to declare and initialize a reference to a NestedClass object?

(a) NormalClass.NestedClass myNC = new NormalClass.NestedClass();
(b) NestedClass myNC = new NormalClass().new NestedClass();
(c) NestedClass myNC = new NormalClass.NestedClass();

7.6) The following is an outline of code for top-level class. Assume that both classes have correct constructors taking no parameters.

```
class BaseClass {
    static class NestedClass {
        // methods and variables of NestedClass
    }
    // methods and variables of BaseClass
}
```

Which of the following code fragments shows the correct way to declare and initialize a reference to a NestedClass object?

(a) BaseClass.NestedClass myNC = new BaseClass.NestedClass();
(b) NestedClass myNC = new BaseClass().new NestedClass();
(c) NestedClass myNC = new BaseClass.NestedClass();
(d) BaseClass.NestedClass nbc = new BaseClass().new NestedClass();

7.7) Which of the following statements about this code are true?

```
public class Morecombe {
    public static void main(String argv[]){
        Morecombe m = new Morecombe();
        m.go(new Turing(){});
    }
    public void go(Turing t){
        t.start();
      }
}
class Turing extends Thread {
    public void run(){
        for(int i =0; i < 2; i++){
            System.out.println(i);
        }
    }
}
```

(a) Compilation error due to malformed parameter to go method.
(b) Compilation error, class Turing has no start method.
(c) Compilation and output of 0 followed by 1.
(d) Compilation but runtime error.

7.8) What will be the result of attempting to compile and run the following code?

```
public class MyClass {
public static void main(String[] args) {
Outer objRef = new Outer();
System.out.println(objRef.createInner().getSecret());
}
}
class Outer {
private int secret;
Outer() { secret = 123; }
class Inner {
int getSecret() { return secret; }
}
Inner createInner() { return new Inner(); }
}
```

Select the one correct answer.

a. The code will fail to compile because the classI nner cannot be declared within the class Outer.
b. The code will fail to compile because the method createInner() cannot be allowed to pass objects of the class Inner to methods outside of the class Outer.
c. The code will fail to compile because thes ecret field is not accessible from the method getSecret().
d. The code will fail to compile because the method getSecret() is not visible from the main() method in the class MyClass.
e. The code will compile without error and will print1 23 when run.

7.9) Which statements are true about nested classes? Select the two correct answers.
a. An instance of a static member class has an inherent outer instance.
b. A static member class can contain non-static fields.
c. A static member interface can contain non-static fields.
d. A static member interface has an inherent outer instance.
e. For each instance of the outer class, there can exist many instances of a non-static member class.

7.10) What will be the result of attempting to compile and run the following code?

```
public class MyClass {
public static void main(String[] args) {
State st = new State();
System.out.println(st.getValue());
State.Memento mem = st.memento();
st.alterValue();
System.out.println(st.getValue());
mem.restore();
System.out.println(st.getValue());
}
public static class State {
protected int val = 11;
int getValue() { return val; }
void alterValue() { val = (val + 7) % 31; }
Memento memento() { return new Memento(); }
class Memento {
int val;
Memento() { this.val = State.this.val; }
void restore() { ((State) this).val = this.val; }
}
}
}
```

Select the one correct answer.
a. The code will fail to compile since the static main() method tries to create a new instance of the static member class State.
b. The code will fail to compile since the declaration of class State.Memento is not accessible from the main() method.
c. The code will fail to compile since the non-static member class Memento declares a field with the same name as a field in the outer class State.
d. The code will fail to compile since theS tate.this.val expression in the Memento constructor is invalid.
e. The code will fail to compile since the ((State) this).val expression in the method restore() of the class Memento is invalid.
f. The program compiles without error and prints1 1, 18, and 11 when run.

7.11) What will be the result of attempting to compile and run the following program?

```
public class Nesting {
public static void main(String[] args) {
B.C obj = new B().new C();
}
}
class A {
int val;
A(int v) { val = v; }
}
```

```
class B extends A {
int val = 1;
B() { super(2); }
class C extends A {
int val = 3;
C() {
super(4);
System.out.println(B.this.val);
System.out.println(C.this.val);
System.out.println(super.val);
}
}
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will compile without error and print2 , 3, and 4, in that order, when run.
c. The program will compile without error and print1 , 4, and 2, in that order, when run.
d. The program will compile without error and print1 , 3, and 4, in that order, when run.
e. The program will compile without error and print3 , 2, and 1, in that order, when run.

7.12) Which statements are true about the following program?

```
public class Outer {
public void doIt() {
}
public class Inner {
public void doIt() {
}
}
public static void main(String[] args) {
new Outer().new Inner().doIt();
}
}
```

Select the two correct answers.

a. The doIt() method in the Inner class overrides the doIt() method in the Outer class.
b. The doIt() method in the Inner class overloads the doIt() method in the Outer class.
c. The doIt() method in the Inner class hides the doIt() method in the Outer class.
d. The full name of the Inner class is Outer.Inner.
e. The program will fail to compile.

7.13) Which statement is true? Select the one correct answer.

a. Non-static member classes must have either default orp ublic accessibility.
b. All nested classes can declare static member classes.
c. Methods in all nested classes can be declared static.
d. All nested classes can be declared static.
e. Static member classes can contain non-static methods.

7.14) Given the declaration

```
interface IntHolder { int getInt(); }
which of the following methods are valid?
//----(1)----
IntHolder makeIntHolder(int i) {
return new IntHolder() {
public int getInt() { return i; }
};
}
//----(2)----
IntHolder makeIntHolder(final int i) {
return new IntHolder {
public int getInt() { return i; }
};
}
//----(3)----
```

```
IntHolder makeIntHolder(int i) {
class MyIH implements IntHolder {
public int getInt() { return i; }
}
return new MyIH();
}
//----(4)----
IntHolder makeIntHolder(final int i) {
class MyIH implements IntHolder {
public int getInt() { return i; }
}
return new MyIH();
}
//----(5)----
IntHolder makeIntHolder(int i) {
return new MyIH(i);
}
static class MyIH implements IntHolder {
final int j;
MyIH(int i) { j = i; }
public int getInt() { return j; }
}
```

Select the two correct answers.

a. The method labeled (1).
b. The method labeled (2).
c. The method labeled (3).
d. The method labeled (4).
e. The method labeled (5).

7.15) Which statements are true? Select the two correct answers.

a. No other static members, exceptf inal static fields, can be declared within a non-static member class.
b. If a non-static member class is nested within a class named Outer, then methods within the non-static member class must use the prefix Outer.this to access the members of the class Outer.
c. All fields in any nested class must be declared final.
d. Anonymous classes cannot have constructors.
e. If objRef is an instance of any nested class within the class Outer, then the expression (objRef instanceof Outer) will evaluate to true.

7.16) Which statement is true? Select the one correct answer.

a. Top-level classes can be declared static.
b. Classes declared as members of top-level classes can be declared static.
c. Local classes can be declared static.
d. Anonymous classes can be declared static.
e. No classes can be declared static.

Nested, Inner Classes & Interfaces Answers:

7.1) (d)
This looks like a question about inner classes but it is also a reference to the fact that the main method is static and thus you cannot directly access a non static method. The line causing the error could be fixed by changing it to say
Inner i = new Outer().new Inner();
Then the code would compile and run producing the output "Inner"

7.2) (a), (d)
A static inner class is also sometimes known as a top level nested class. There is some debate if such a class should be called an inner class. I tend to think it should be on the basis that it is created inside the opening braces of another class. How could an anonymous class have a constructor?. Remember a constructor is a method with no return type and the same name as the class. Inner classes may be defined as private

7.3) (a), (d)
A class within a method can only see final variables of the enclosing method. However it the normal visibility rules apply for variables outside the enclosing method.

7.4) (c), (d)
The import statement allows you to use a class directly instead of fully qualifying it with the full package name, adding more classess with the import statement does not cause a runtime performance overhad. I assure you this is true. An inner class can be defined with the private modifier.

7.5) (a)
(a) shows both correct reference variable declaration and a correct constructor. (b), (c) are incorrect declaration of the reference variable because the name of a nested class starts with the enclosing class name.

7.6) (d)
(d) shows both correct reference variable declaration and a correct constructor. This statement creates a BaseClass object, which the NestedClass must have. (b), (c) are incorrect declaration of the reference variable because the name of a nested class starts with the enclosing class name. Answer (a) does not create a BaseClass object for the NestedClass to be associated with.

7.7) (c)
The creation of an anonymous class as a parameter to go is fairly strange as you would expect it to override a method in its parent class (Turing). You don't have to though. The fact that class Turing extends Thread means the anonymous instance that is passed to go has a start method which then calls the run method.

7.8) (e)
The code will compile without error and will print 123 when run. An instance of the Outer outer will be created and the field secret will be initialized to 123. A call to the createInner() method will return a reference to a newly created Inner instance. This object is an instance of a non-static member class and is associated with the outer instance. This means that an object of a non-static member class has access to the members within the outer instance. Since the Inner class is nested in the class containing the field secret, this field is accessible to the Inner instance, even though the field secret is declared private.

7.9) (b) and (e)
A static member class is in many respects like a top-level class, and can contain non-static fields. Instances of non-static member classes are created in the context of an outer instance. The outer instance is inherently associated with the inner instance. Several inner class instances can be created and associated with the same outer instance. Static classes do not have any inherent outer instance. A static member interface, just like top-level interfaces, cannot contain non-static fields. Nested interfaces are always static.

7.10) (e)

The code will fail to compile, since the expression ((State) this).val in the method restore() of the class Memento is invalid. The correct way to access the field val in the class State, which is hidden by the field val in the class Memento, is to use the expression State.this.val. Other than that, there are no problems with the code.

7.11) (d)
The program will compile without error, and will print 1, 3, 4, in that order, when run. The expression B.this.val will access the value 1 stored in the field val of the (outer) B instance associated with the (inner) C object denoted by the reference obj. The expression C.this.val will access the value 3 stored in the field val of the C object denoted by the reference obj. The expression super.val will access the field val from A, the superclass of C.

7.12) (c) and (d)
The class Inner is a non-static member class of the Outer class, and its full name is Outer.Inner. The Inner class does not inherit from the Outer class. The method named doIt is, therefore, neither overridden nor overloaded. Within the scope of the Inner class, the doIt() method of the Outer class is hidden by the doIt() method of the Inner class.

7.13) (e)
Non-static member classes, unlike top-level classes, can have any accessibility modifier. Static member classes can only be declared in top-level or nested static member classes and interfaces. Only static member classes can be declared static. Declaring a class static only means that instances of the class are created without having an outer instance. This has no bearing on whether the members of the class can be static or not.

7.14) (d) and (e)
The methods labeled (1) and (3) will not compile, since the non-final parameter i is not accessible from within the inner class. The syntax of the anonymous class in the method labeled (2) is not correct, as the parameter list is missing.

7.15) (a) and (d)
No other static members, except final static fields, can be declared within a non-static member class. Members in outer instances are directly accessible using simple names (provided they are not hidden). Fields in nested static member classes need not be final. Anonymous classes cannot have constructors, since they have no names. Nested classes define distinct types from the enclosing class, and the instanceof operator does not take the type of the outer instance into consideration.

7.16) (b)
Classes can be declared as members of top-level classes. Such a class is a static member class if it is declared static, otherwise, it is a non-static member class. Top-level classes, local classes, and anonymous classes cannot be declared static.

Multi-Threading Mock Questions

8.1) What will happen when you attempt to compile and run the following code?

```
public class Bground extends Thread{
public static void main(String argv[]){
                Bground b = new Bground();
                b.run();
        }
        public void start(){
                for (int i = 0; i <10; i++){
                        System.out.println("Value of i = " + i);
                }
        }
}
```

(a) A compile time error indicating that no run method is defined for the Thread class.
(b) A run time error indicating that no run method is defined for the Thread class.
(c) Clean compile and at run time the values 0 to 9 are printed out.
(d) Clean compile but no output at runtime.

8.2) What can cause a thread to stop executing?
(a) The program exits via a call to System.exit(0);
(b) Another thread is given a higher priority.
(c) A call to the thread's stop method.
(d) A call to the halt method of the Thread class.

8.3) Which statement is true of the following code?

```
public class Agg {
    public static void main(String argv[]){
        Agg a = new Agg();
        a.go();
    }
    public void go(){
        DSRoss ds1 = new DSRoss("one");
        ds1.start();
    }
}

class DSRoss extends Thread {
    private String sTname = "";
    DSRoss(String s){
        sTname = s;
    }
    public void run(){
        notwait();
        System.out.println("finished");
    }
    public void notwait(){
        while(true) {
            try {
                System.out.println("waiting");
                wait();
            } catch(InterruptedException ie){
            }
            System.out.println(sTname);
            notifyAll();
        }
    }
}
```

(a) It will cause a compile time error.
(b) Compilation and output of "waiting".
(c) Compilation and output of "waiting" followed by "finished".
(d) Runtime error, an exception will be thrown.

8.4) What will happen when you attempt to compile and run the following code?

```
public class Holt extends Thread {
    private String sThreadName;
    public static void main(String argv[]){
        Holt h = new Holt();
        h.go();
    }
    Holt() {
    }
    Holt(String s) {
        sThreadName = s;
    }
    public String getThreadName() {
        return sThreadName;
    }
    public void go() {
        Holt first = new Holt("first");
        first.start();
        Holt second = new Holt("second");
        second.start();
    }
    public void start(){
        for(int i = 0; i < 2; i ++){
            System.out.println(getThreadName() +i);
            try {
                Thread.sleep(100);
            } catch(InterruptedException e) {
                System.out.println(e.getMessage());}
            }
        }
    }
```

(a) Compile time error.
(b) Output of first0, second0, first0, second1.
(c) Output of first0, first1, second0, second1.
(d) Runtime error.

8.5) What will happen when you attempt to compile and run the following code?

```
class Background implements Runnable{
    int i = 0;
    public int run(){
         while (true) {
             i++;
             System.out.println("i="+i);
         }
         return 1;
     }
}//End class
```

(a) It will compile and the run method will print out the increasing value of i.
(b) It will compile and calling start will print out the increasing value of i.
(c) The code will cause an error at compile time.
(d) Compilation will cause an error because while cannot take a parameter of true.

8.6) Which statement is true of the following code?

```
public class Rpcraven {
    public static void main(String argv[]){
        Pmcraven pm1 = new Pmcraven("one");
        pm1.run();
        Pmcraven pm2 = new Pmcraven("two");
        pm2.run();
    }
}
class Pmcraven extends Thread {
```

```
        private String sTname="";
        Pmcraven(String s) {
            sTname = s;
        }
        public void run(){
              for(int i =0; i < 2 ; i++){
                try {
                    sleep(1000);
                }catch(InterruptedException e){
                }
                yield();
                System.out.println(sTname);
            }
        }
    }
```

(a) Compile time error, class Rpcraven does not import java.lang.Thread
(b) Output of One One Two Two
(c) Output of One Two One Two
(d) Output of One Two One Two

8.7) You are creating a class that extends Object and implements Runnable. You have already written a run method for this class, and you need a way to create a thread and have it execute the run method, Which of these start methods should you use?
(a) public void start() { new Thread(this).start(); }
(b) public void start() { Thread myT = new Thread(); myT.start(); }
(c) public void start() { Thread myT = new Thread(this); myT.run(); }

8.8) Java Thread A is attached to an object B, which is responsible for writing data to a file. After writing data, Thread A calls the following method in object B, where it waits until more data is available.

```
    private void synchronized waitForData() {    // (1)
        try {                                    // (2)
            wait();                              // (3)
        } catch (InterruptedException ex) {      // (4)
        }
    }
```

Another Thread, executing a method in another object C, needs to wake up Thread A. Assuming that object C has references to both A and B, select all of the following code fragments that would cause Thread A to exit the waitForData method.
(a) A.interrupt();
(b) synchronized(A) { A.notifyAll(); }
(c) synchronized(B) { B.notifyAll(); }
(d) A.resume();

8.9) You have created a TimeOut class as an extension of Thread, the purpose of which is to print a "Time's Up" message if the Thread is not interrupted within 10 seconds of being started. Here is the run method that you have coded:

```
    public void run() {
        System.out.println("Start!");
        try {
            Thread.sleep(10000);
            System.out.println("Time's Up!");
        } catch (InterruptedException e) {
            System.out.println("Interrupted!");
        }
    }
```

Given that a program creates and starts a TimeOut object, which of the following statements is true?
(a) Exactly 10 seconds after the start method is called, "Time's Up!" will be printed.
(b) Exactly 10 seconds after the "Start!" is printed, "Time's Up!" will be printed.
(c) The delay between "Start!" being printed and "Time's Up!" will be 10 seconds plus or minus one tick of the system clock.

(d) If "Time's Up!" is printed, you can be sure that at least 10 seconds have elapsed since "Start!" was printed.

8.10) Any class that implements the Runnable interface has to provide the implementation for the following methods

```
public void start();
public void run();
```

(a)True.
(b)False.

8.11) True or false ?
A thread that has called the wait() method of an object still owns the lock of the object.
(a)True
(b)False

8.12) A number of threads of the same priority have relinquished the lock on a monitor and are in a waiting state after having called the wait() method of the object. A new thread enters the monitor and calls the notifyAll() method of the meonitor. Which of these threads will be the first one to resume?
(a)The thread that has been waiting the longest.
(b)The thread that was the last one to to exit the monitor.
(c)You can never be sure which thread will get to run first.
(d)The the first thread that called the wait() method

8.13) Which of these are valid contructors of a Thread object.
(a)public Thread(Object obj)
(b)public Thread(String name)
(b)public Thread(Runnable trgt)
(d)public Thread(ThreadGroup grp, Runnable trgt, String name)
(e)public Thread(ThreadGroup grp, Object ob)

8.14) If you call the interrupted() method of a thread object twice the second call will always return false.
(a)True
(b)False

8.15) If you call the isInterrupted() method of a thread object twice the second call will always return false.
(a)True
(b)False

8.16) Which of the following are methods of the Thread class.
(a)public void run()
(b)public void start()
(c)public void exit()
(d)public final void setAccess()
(e)public final void setPriority(int priNbr)
(f)public final int getPriority()

8.17) Consider the following class

```
public class Test implements Runnable{
      public void run(){}
}
```

True or False ? Creating an instance of this class and calling its run() method will spawn a new thread.
(a)True
(b)False

8.18) True or false? A Thread object has a method called notify().
(a)False
(b)True

8.19) Calling the destroy() method of a thread object relases all the locks held by the thread ?
(a)True

(b)False

8.20) Which is the correct way to start a new thread? Select the one correct answer.
a. Just create a new Thread object. The thread will start automatically.
b. Create a new Thread object and call the method begin().
c. Create a new Thread object and call the method start().
d. Create a new Thread object and call the method run().
e. Create a new Thread object and call the method resume().

8.21) When extending the Thread class to provide a thread's behavior, which method should be overridden? Select the one correct answer.
a. begin()
b. start()
c. run()
d. resume()
e. behavior()

8.22) Which statements are true? Select the two correct answers.
a. The class Thread is abstract.
b. The class Thread implements Runnable.
c. Classes implementing the Runnable interface must define a method named start.
d. Calling the method run() on an object implementing Runnable will create a new thread.
e. A program terminates when the last non-daemon thread ends.

8.23) What will be the result of attempting to compile and run the following program?

```
public class MyClass extends Thread {
public MyClass(String s) { msg = s; }
String msg;
public void run() {
System.out.println(msg);
}
public static void main(String[] args) {
new MyClass("Hello");
new MyClass("World");
}
}
```

Select the one correct answer.
a. The program will fail to compile.
b. The program will compile without errors and will print Hello and World, in that order, every time the program is run.
c. The program will compile without errors and will print a never-ending stream oHf ello and World.
d. The program will compile without errors and will print Hello and World when run, but the order is unpredictable.
e. The program will compile without errors and will simply terminate without any output when run.

8.24) Given the following program, which statements are guaranteed to be true?

```
public class ThreadedPrint {
static Thread makeThread(final String id, boolean daemon) {
Thread t = new Thread(id) {
public void run() {
System.out.println(id);
}
};
t.setDaemon(daemon);
t.start();
return t;
}
public static void main(String[] args) {
Thread a = makeThread("A", false);
Thread b = makeThread("B", true);
System.out.print("End\n");
```

```
}
}
```

Select the two correct answers.

a. The letter A is always printed.
b. The letter B is always printed.
c. The letter A is never printed after End.
d. The letter B is never printed after End.
e. The program might print B, End and A, in that order.

8.25) Which statement is true? Select the one correct answer.

a. No two threads can concurrently execute synchronized methods on the same object.
b. Methods declared synchronized should not be recursive, since the object lock will not allow new invocations of the method.
c. Synchronized methods can only call other synchronized methods directly.
d. Inside a synchronized method, one can assume that no other threads are currently executing any other methods in the same class.

8.26) Given the following program, which statement is true?

```
public class MyClass extends Thread {
static Object lock1 = new Object();
static Object lock2 = new Object();
static volatile int i1, i2, j1, j2, k1, k2;
public void run() { while (true) { doit(); check(); } }
void doit() {
synchronized(lock1) { i1++; }
j1++;
synchronized(lock2) { k1++; k2++; }
j2++;
synchronized(lock1) { i2++; }
}
void check() {
if (i1 != i2) System.out.println("i");
if (j1 != j2) System.out.println("j");
if (k1 != k2) System.out.println("k");
}
public static void main(String[] args) {
new MyClass().start();
new MyClass().start();
}
}
```

Select the one correct answer.

a. The program will fail to compile.
b. One cannot be certain whether any of the letters i, j, and k will be printed during execution.
c. One can be certain that none of the letters i, j, and k will ever be printed during execution.
d. One can be certain that the letters i and k will never be printed during execution.
e. One can be certain that the letter k will never be printed during execution.

8.27) Which one of these events will cause a thread to die? Select the one correct answer.

a. The method sleep() is called.
b. The method wait() is called.
c. Execution of the start() method ends.
d. Execution of the run() method ends.
e. Execution of the thread's constructor ends.

8.28) Which statements are true about the following code?

```
public class Joining {
static Thread createThread(final int i, final Thread t1) {
Thread t2 = new Thread() {
public void run() {
System.out.println(i+1);
try {
t1.join();
```

```
} catch (InterruptedException e) {
}
System.out.println(i+2);
}
};
System.out.println(i+3);
t2.start();
System.out.println(i+4);
return t2;
}
public static void main(String[] args) {
createThread(10, createThread(20, Thread.currentThread()));
}
}
```

Select the two correct answers.

a. The first number printed is 13.
b. The number 14 is printed before the number 22.
c. The number 24 is printed before the number 21.
d. The last number printed is 12.
e. The number 11 is printed before the number 23.

8.29) What can be guaranteed by calling the method yield()? Select the one correct answer.

a. All lower priority threads will be granted CPU time.
b. The current thread will sleep for some time while some other threads run.
c. The current thread will not continue until other threads have terminated.
d. The thread will wait until it is notified.
e. None of the above.

8.30) Where is the notify() method defined? Select the one correct answer.

a. Thread
b. Object
c. Applet
d. Runnable

8.31) How can the priority of a thread be set? Select the one correct answer.

a. By using the setPriority() method in the class Thread.
b. By passing the priority as a parameter to the constructor of the thread.
c. Both of the above.
d. None of the above.

8.32) Which statements are true about locks? Select the two correct answers.

a. A thread can hold more than one lock at a time.
b. Invoking wait() on a Thread object will relinquish all locks held by the thread.
c. Invoking wait() on an object whose lock is held by the current thread will relinquish the lock.
d. Invoking notify() on a object whose lock is held by the current thread will relinquish the lock.
e. Multiple threads can hold the same lock at the same time.

8.33) What will be the result of invoking the wait() method on an object without ensuring that the current thread holds the lock of the object? Select the one correct answer.

a. The code will fail to compile.
b. Nothing special will happen.
c. An IllegalMonitorStateException will be thrown if the wait() method is called while the current thread does not hold the lock of the object.
d. The thread will be blocked until it gains the lock of the object.

8.34) Which of these are plausible reasons why a thread might be alive, but still not be running? Select the four correct answers.

a. The thread is waiting for some condition as a result of aw ait() call.
b. The execution has reached the end of the run() method.
c. The thread is waiting to acquire the lock of an object in order to execute a certain method on that object.

d. The thread does not have the highest priority and is currently not executing.
e. The thread is sleeping as a result of a call to thes leep() method.

Multi-Threading Answers

8.1) (d)
This is a bit of a sneaky one as I have swapped around the names of the methods you need to define and call when running a thread. If the for loop were defined in a method called public void run() and the call in the main method had been to b.start()
The list of values from 0 to 9 would have been output.

8.2) (a), (b), (c)
Java threads are somewhat platform dependent and you should be carefull when making assumptions about Thread priorities. On some platforms you may find that a Thread with higher priorities gets to "hog" the processor.

8.3) (d)
A call to wait/notify must be within synchronized code. With JDK1.2 this code throws the error message
java.lang.IllegalMonitorStateException: current thread not owner
at java.lang.Object.wait(Native Method)
at java.lang.Object.wait(Object.java:424)
at DSRoss.notwait(Compiled Code)
at DSRoss.run(Agg.java:21)

8.4) (c)
Note that this code overrides and calls the start method. If you wished to get the output mixed you would need to override the run method but call the start method.

8.5) (c)
The error is caused because run should have a void not an int return type.
Any class that is implements an interface must create a method to match all of the methods in the interface. The Runnable interface has one method called run that has a void return type.The sun compiler gives the error
Method redefined with different return type: int run() was defined as void run();

8.6) (b)
Answer (c) would would be true if the code called the start method instead of the run method (well it is on my Windows machine anyway, I'm not sure it would be for ever implementation of Java Threads). If you call the run method directly it just acts as any other method and does not return to the calling code until it has finished executing.

8.7) (a)
Answer (a) is ok, it does not matter that there is no reference to the thread in the start method. Answer (b) is incorrect because the new Thread is not attached to the Runnable object so it cannot find the run method. Instead, the default run in the Thread class will be executed. Answer (c) is incorrect because the Thread that is executing the start method calls run in the Thread class. The myT Thread is not started.

8.8) (a), (c)
Answer (a) is correct because it will generate an InterruptedException, bringing thread A out of the wait. Because line (4) catches this exception, the thread exits the waitForData method normally. Answer (c) is correct because it removes all Threads waiting for object B from the wait list, including thread A. Answer (b) is incorrect because it refers to the thread A, not to the object B for which the thread is waiting. Answer (d) is incorrect because the resume method works only with suspended threads and is a deprecated method.

8.9) (d)
It is the only statement that can be made with confidence. Answer (a), (b), (c) are all incorrect because the expiration of a sleep timer does not guarantee that a Thread will run – only that it can run.

8.10) (b)

8.11) (b)

8.12) (c)

8.13) (b), (c), (d)

8.14) (a)

8.15) (b)

8.16) (a), (b), (e), (f)

8.17) (b)

8.18) (b)
The Thread class has a method notify() inherited from Object.

8.19) (b)

8.20) (c)
Create a new Thread object and call the method start(). The call to the start() method will return immediately, and the thread will start executing the run() method asynchronously.

8.21) (c)
When extending the Thread class, the run() method should be overridden to provide the code executed by the thread. This is analogous to implementing the run() method of the Runnable interface.

8.22) (b) and (e)
The Thread class implements the Runnable interface and is not abstract. A program terminates when the last non-daemon thread ends. The Runnable interface has a method named run, but the interface does not dictate that implementations must define a method named start. Calling the run() method on a Runnable object does not necessarily create a new thread. Method run() is the method executed by a thread. Instances of the Thread class must be created to spawn new threads.

8.23) (e)
The program will compile without errors, and will simply terminate without any output when run. Two thread objects will be created, but they will never be started. The start() method must be called on the thread objects to make the threads execute the run() method asynchronously.

8.24) (a) and (e)
Because the exact behavior of the scheduler is undefined, the text A, B, and End can be printed in any order. The thread printing B is a daemon thread, which means that the program may terminate before the thread manages to print the letter.

8.25) (a)
No two threads can concurrently execute synchronized methods on the same object. This does not prevent one thread from executing a non-synchronized method while another thread executes a synchronized method on the same object. The synchronization mechanism in Java acts like recursive semaphores, which means that during the time a thread owns the lock, it may enter and re-enter any region of code associated with the lock.

8.26) (b)
One cannot be certain whether any of the letters i, j, and k will be printed during execution. For each invocation of the doit() method, each variable pair is incremented and their values are always equal when the method returns. The only way a letter could be printed would be if the method check() was executed between the time the first and the second variable were incremented. Since the check() method does not depend on owning any lock, it can be executed at any time, and the method doit() cannot protect the atomic nature of its operations by acquiring locks.

8.27) (d)

A thread dies when the execution of the run() method ends. The call to the start() method is asynchronous, that is, it returns immediately, and it enables the thread for running. Calling the sleep() or wait() methods will only block the thread temporarily.

8.28) (b) and (d)
The inner createThread() call is evaluated first, and will print 23 as the first number. The last number the main thread prints is 14. After the main thread ends, the thread created by the inner createdThread() completes its join() call and prints 22. After this thread ends, the thread created by the outer createThread() call completes its join() call and prints the number 12 before the program terminates.

8.29) (e)
The exact behavior of the scheduler is not defined. There is no guarantee that a call to the yield() method will grant other threads use of the CPU.

8.30) (b)
The notify() method is defined in the Object class.

8.31) (a)
The priority of a thread is set by calling the setPriority() method in the Thread class. No Thread constructor accepts a priority level as an argument.

8.32) (a) and (c)
A thread can hold multiple locks; for example, by nesting synchronized blocks. Invoking the wait() method on an object whose lock is held by the current thread will relinquish the lock for the duration of the call. The notify() method does not relinquish any locks.

8.33) (c)
An IllegalMonitorStateException will be thrown if the wait() method is called when the current thread does not hold the lock of the object.

8.34) (a), (c), (d), and (e)
The thread terminates once the run() method completes execution.

Fundamental Classes Mock Questions

9.1) What is the output of the following:

```
StringBuffer sb1 = new StringBuffer("Amit");
StringBuffer sb2 = new StringBuffer("Amit");
String ss1 = "Amit";  System.out.println(sb1==sb2);
System.out.println(sb1.equals(sb2));
System.out.println(sb1.equals(ss1));
System.out.println("Poddar".substring(3));
```

a) false false false dar
b) false true false Poddar
c) Compiler Error
d) true true false dar

9.2) Which of the following will output -4.0 ?

(a) System.out.println(Math.floor(-4.7));
(b) System.out.println(Math.round(-4.7));
(c) System.out.println(Math.ceil(-4.7));
(d) System.out.println(Math.min(-4.7));

9.3) What will be the result when you attempt to compile this program?

```
public class Rand {
    public static void main(String argv[]){
        int iRand;
        iRand = Math.random();
        System.out.println(iRand);
    }
}
```

(a) Compile time error referring to a cast problem.
(b) A random number between 1 and 10.
(c) A random number between 0 and 1
(d) A compile time error about random being an unrecognised method.

9.4) What will be the result of compiling and running the given program?
Select one correct answer.

```
1  class Q2
2  {
3     public static void main(String arg[])
4     {
5        StringBuffer s[]={"A","B"};
6        System.out.println(s[0]+","+s[1]);
7     }
8  }
```

(a) Program compiles correctly and print A,B when executed.
(b) Compile time error.
(c) Run time error.

9.5) What will be the value of str after line no.7?
Enter your answer in the given field. Don't use any extra signs like ("")quotes or any string like "str=".
For example, if your answer is the string "XYZ" you just write XYZ

```
1  class string
2  {
3     public static void main(String ds[])
4     {
5        String str="A";
6        str.concat("B");
7        str+="C";
8     }
9  }
```

9.6) What will be the result of compiling and running the given program?

Select one correct answer.

```
1  class strings
2  {
3     public static void main(String arg[])
4     {
5          if("String".indexOf("S",-10) ==0)
6          {
7              System.out.println("Equal");
8          }
9          else
10         {
11             System.out.println("Not Equal");
12         }
13    }
14 }
```

(a) Compile time error as we passed negative value in indexOf() method.
(b) Run time error as we passed negative value in indexOf() method.
(c) Program compiles and prints Equal when executed.
(d) Program compiles and prints Not Equal when executed.

9.7) What is the return type of the hashCode() method in the Object class? Select the one correct answer.
a. String
b. int
c. long
d. Object
e. Class

9.8) Which statement is true? Select the one correct answer.
a. If the references x and y denote two different objects, then the expressionx .equals(y) is always false.
b. If the references x and y denote two different objects, then the expression (x.hashCode() == y.hashCode()) is always false.
c. The hashCode() method in the Object class is declared final.
d. The equals() method in the Object class is declared final.
e. All arrays have a method namedc lone.

9.9) Which exception can the clone() method of the Object class throw? Select the one correct answer.
a. CloneNotSupportedException
b. NotCloneableException
c. IllegalCloneException
d. NoClonesAllowedException

9.10) Which of the following are wrapper classes? Select the three correct answers.
a. java.lang.Void
b. java.lang.Int
c. java.lang.Boolean
d. java.lang.Long
e. java.lang.String

9.11) Which of the following classes do not extend the java.lang.Number class? Select the two correct answers.
a. java.lang.Float
b. java.lang.Byte
c. java.lang.Character
d. java.lang.Boolean
e. java.lang.Short

9.12) Which of these classes define immutable objects? Select the three correct answers.
a. Character
b. Byte
c. Thread

d. Short
e. Object

9.13) Which of these classes have a one-parameter constructor taking a string? Select the two correct answers.
a. Void
b. Integer
c. Boolean
d. Character
e. Object

9.14) Which of the wrapper classes have a booleanValue() method? Select the one correct answer.
a. All wrapper classes.
b. All wrapper classes except Void.
c. All wrapper classes that also implement the compareTo() method.
d. All wrapper classes extending Number.
e. Only the class Boolean.

9.15) Which statements are true about wrapper classes? Select the two correct answers.
a. String is a wrapper class.
b. Double has a compareTo() method.
c. Character has a intValue() method.
d. Byte extends Number.

9.16) Given the following program, which lines will print 11 exactly?

```
class MyClass {
public static void main(String[] args) {
double v = 10.5;
System.out.println(Math.ceil(v)); // (1)
System.out.println(Math.round(v)); // (2)
System.out.println(Math.floor(v)); // (3)
System.out.println((int) Math.ceil(v)); // (4)
System.out.println((int) Math.floor(v)); // (5)
}
}
```

Select the two correct answers.
a. The line labeled (1).
b. The line labeled (2).
c. The line labeled (3).
d. The line labeled (4).
e. The line labeled (5).

9.17) Which method is not defined in the Math class? Select the one correct answer.
a. double tan2(double)
b. double cos(double)
c. int abs(int a)
d. double ceil(double)
e. float max(float, float)

9.18) What is the return type of the method round(float) from the Math class? Select the one correct answer.
a. int
b. float
c. double
d. Integer
e. Float

9.19) What is the return type of the method ceil(double) from the Math class? Select the one correct answer.
a. int

b. float
c. double
d. Integer
e. Double

9.20) What will the following program print when run?

```
public class Round {
public static void main(String[] args) {
System.out.println(Math.round(-0.5) + " " + Math.round(0.5));
}
};
```

Select the one correct answer.

a. 0 0
b. 0 1
c. -1 0
d. -1 1
e. None of the above.

9.21) Which statements are true about the expression ((int)(Math.random()*4))? Select the three correct answers.

a. It may evaluate to a negative number.
b. It may evaluate to the number 0.
c. The probability of it evaluating to the number 1 or the number 2 is the same.
d. It may evaluate to the number 3.
e. It may evaluate to the number 4.

9.22) Which of the following operators cannot be used in conjunction with aS tring object? Select the two correct answers.

a. +
b. -
c. +=
d. .
e. &

9.23) Which expression will extract the substring "kap" from a string defined by String str = "kakapo"? Select the one correct answer.

a. str.substring(2, 2)
b. str.substring(2, 3)
c. str.substring(2, 4)
d. str.substring(2, 5)
e. str.substring(3, 3)

9.24) What will be the result of attempting to compile and run the following code?

```
class MyClass {
public static void main(String[] args) {
String str1 = "str1";
String str2 = "str2";
String str3 = "str3";
str1.concat(str2);
System.out.println(str3.concat(str1));
}
}
```

Select the one correct answer.

a. The code will fail to compile since the expression str3.concat(str1) will not result in a valid argument for the println() method.
b. The program will print str3str1str2 when run.
c. The program will print str3 when run.
d. The program will print str3str1 when run.
e. The program will print str3str2 when run.

9.25) What function does the trim() method of the String class perform? Select the one correct answer.

a. It returns a string where the leading white space of the original string has been removed.
b. It returns a string where the trailing white space of the original string has been removed.
c. It returns a string where both the leading and trailing white space of the original string has been removed.
d. It returns a string where all the white space of the original string has been removed.
e. None of the above.

9.26) Which statements are true? Select the two correct answers.
a. String objects are immutable.
b. Subclasses of the String class can be mutable.
c. All wrapper classes are declared final.
d. All objects have a public method named clone().
e. The expression ((new StringBuffer()) instanceof String) is always true.

9.27) Which of these expressions are legal? Select the four correct answers.
a. "co".concat("ol")
b. ("co" + "ol")
c. ('c' + 'o' + 'o' + 'l')
d. ("co" + new String('o' + 'l'))
e. ("co" + new String("co"))

9.28) What will be the result of attempting to compile and run the following code?

```
public class RefEq {
public static void main(String[] args) {
String s = "ab" + "12";
String t = "ab" + 12;
String u = new String("ab12");
System.out.println((s==t) + " " + (s==u));
}
}
```

Select the one correct answer.
a. The code will fail to compile.
b. The program will print false false when run.
c. The program will print false true when run.
d. The program will print true false when run.
e. The program will print true true when run.

9.29) Which of these parameter lists have a corresponding constructor in theS tring class? Select the three correct answers.
a. ()
b. (int capacity)
c. (char[] data)
d. (String str)

9.30) Which method is not defined in the String class? Select the one correct answer.
a. trim()
b. length()
c. concat(String)
d. hashCode()
e. reverse()

9.31) Which statement concerning the charAt() method of the String class is true? Select the one correct answer.
a. The charAt() method takes a char value as an argument.
b. The charAt() method returns a Character object.
c. The expression ("abcdef").charAt(3) is illegal.
d. The expression "abcdef".charAt(3) evaluates to the character 'd'.
e. The index of the first character is 1.

9.32) Which expression will evaluate to true? Select the one correct answer.

a. "hello: there!".equals("hello there")
b. "HELLO THERE".equals("hello there")
c. ("hello".concat("there")).equals("hello there")
d. "Hello There".compareTo("hello there") == 0
e. "Hello there".toLowerCase().equals("hello there")

9.33) What will the following program print when run?

```
public class Search {
public static void main(String[] args) {
String s = "Contentment!";
int middle = s.length()/2;
String nt = s.substring(middle-1, middle+1);
System.out.println(s.lastIndexOf(nt, middle));
}
};
```

Select the one correct answer.

a. 2
b. 4
c. 5
d. 7
e. 9
f. 11

9.34) What will be the result of attempting to compile and run the following program?

```
public class MyClass {
public static void main(String[] args) {
String s = "hello";
StringBuffer sb = new StringBuffer(s);
sb.reverse();
if (s == sb) System.out.println("a");
if (s.equals(sb)) System.out.println("b");
if (sb.equals(s)) System.out.println("c");
}
}
```

Select the one correct answer.

a. The code will fail to compile since the constructor of theS tring class is not called properly.
b. The code will fail to compile since the expression (s == sb) is illegal.
c. The code will fail to compile since the expression (s.equals(sb)) is illegal.
d. The program will print c when run.
e. The program will throw a ClassCastException when run.

9.35) What will be the result of attempting to compile and run the following program?

```
public class MyClass {
public static void main(String[] args) {
StringBuffer sb = new StringBuffer("have a nice day");
sb.setLength(6);
System.out.println(sb);
}
}
```

Select the one correct answer.

a. The code will fail to compile since there is no method named setLength in the StringBuffer class.
b. The code will fail to compile since the StringBuffer reference sb is not a legal argument to the println() method.
c. The program will throw a StringIndexOutOfBoundsException when run.
d. The program will print have a nice day when run.
e. The program will print have a when run.
f. The program will print ce day when run.

9.36) Which of these parameter lists have a corresponding constructor in theS tringBuffer class? Select the three correct answers.

a. ()

b. (int capacity)
c. (char[] data)
d. (String str)

9.37) Which method is not defined in the StringBuffer class? Select the one correct answer.
a. trim()
b. length()
c. append(String)
d. reverse()
e. setLength(int)

9.38) What will be the result of attempting to compile and run the following code?

```
public class StringMethods {
public static void main(String[] args) {
String str = new String("eenny");
str.concat(" meeny");
StringBuffer strBuf = new StringBuffer(" miny");
strBuf.append(" mo");
System.out.println(str + strBuf);
}
}
```

Select the one correct answer.
a. The code will fail to compile.
b. The program will print eenny meeny miny mo when run.
c. The program will print meeny miny mo when run.
d. The program will print eenny miny mo when run.
e. The program will print eenny meeny miny when run.

Fundamental Classses Answers

9.1) (a)
No declaration is added, but When I run the example I found that the equals() method in the StringBuffer class is said that it is inherited from the Object class, and I think it didn't override this method, as when I opened String class documentation it didn't said that it inherits the equals() from the Object class, and it gave full description to it. And if this was true and the equals() method is the one inherited by the Object it is false, because it will return true only if the two reference denote the same object.

9.2) (c)
Options (a) and (b) will produce -5 and option 4 will not compile because the min method requires 2 parameters.

9.3) (a)
This is a bit of a sneaky one as the Math.random method returns a pseudo random number between 0 and 1, and thus option (c) is a plausible Answer. However the number returned is a double and so the compiler will complain that a cast is needed to convert a double to an int.

9.4) (b)
Compile time error:Casting needed to convert String to StringBuffer in line no. 5. [Mine opinion this will not work, to make it work ⇒ new StringBuffer("A"), new StringBuffer("B")]

9.5) AC
After concatenating "B" in str the result is not stored so it will not effect at all.Moreover main reason behind this is that Strings are immutable,it always return new object which means if we don't save the results then it will not effect the orignal string at all.

9.6) (c)
We can use negative argument in indexOf() and lastIndexOf() method.In such case it will start searching from zero.

9.7) (b)
The method hashCode() in the Object class returns a hash code value of type int.

9.8) (e)
All arrays are genuine objects and inherit all the methods defined in the Object class, including the clone() method. Neither the hashCode() method nor the equals() method is declared final in the Object() class, and it cannot be guaranteed that implementations of these methods will differentiate between all objects.

9.9) (a)
The clone() method of the Object class will throw a CloneNotSupportedException if the class of the object does not implement the Cloneable interface.

9.10) (a), (c), and (d)
The class java.lang.Void is considered a wrapper class, although it does not wrap any value. There is no class named java.lang.Int, but there is a wrapper class named java.lang.Integer. A class named java.lang.String also exists, but it is not a wrapper class since all strings in Java are objects.

9.11) (c) and (d)
The classes Character and Boolean are non-numeric wrapper classes and they do not extend the Number class. The classes Byte, Short, Integer, Long, Float, and Double are numeric wrapper classes that extend the Number class.

9.12) (a), (b), and (d)
All instances of wrapper classes are immutable.

9.13) (b) and (c)

All instances of wrapper classes except Void and Character have a constructor that accepts a string parameter. The class Object has only a default constructor.

9.14) (e)
While all numeric wrapper classes have the methods byteValue(), doubleValue(), floatValue(), intValue(), longValue(), and shortValue(), only the Boolean class has the booleanValue() method. Likewise, only the Character class has the charValue() method.

9.15) (b) and (d)
String is not a wrapper class. All wrapper classes except Boolean and Void have a compareTo() method. Only the numeric wrapper classes have an intValue() method. The Byte class, like all other numeric wrapper classes, extends the Number class.

9.16) (b) and (d)
The lines labeled (2) and (4) will print 11 exactly, since their expressions will return the int value 11. The expression Math.ceil(v) will return the double value 11.0, which will be printed as 11.0. The expression Math.floor(v) will return 10.0d.

9.17) (a)
The Math class does not have a method named tan2. However, it does have a method named atan2, which converts rectangular coordinates to polar coordinates.

9.18) (a)
The method round(float) will return a value of type int. A round(double) method also exists, which returns a value of type long.

9.19) (c)
The rounding function ceil() will return a value of type double. This is in contrast to the round() methods that will return values of integer types.

9.20) (b)
The value -0.5 is rounded up to 0 and the value 0.5 is rounded up to 1.

9.21) (b), (c), and (d)
The expression will evaluate to one of the numbers 0, 1, 2, or 3. Each number has an equal probability of being returned by the expression.

9.22) (b) and (e)
The operators - and & cannot be used in conjunction with a String object. The operators + and += perform concatenation on strings, and the dot operator accesses members of the String object.

9.23) (d)
The expression str.substring(2, 5) will extract the substring "kap". The method extracts the characters from index 2 to index 4, inclusive.

9.24) (d)
The program will print str3str1 when run. The concat() method will create and return a new String object, which is the concatenation of the current String object and the String object given as an argument. The expression statement str1.concat(str2) creates a new String object, but its reference value is not stored.

9.25) (c)
The trim() method of the String class returns a string where both the leading and the trailing white space of the original string have been removed.

9.26) (a) and (c)
The String class and all wrapper classes are declared final and, therefore, cannot be extended. The clone() method is declared protected in the Object class. String objects are immutable and, therefore, cannot be modified. The classes String and StringBuffer are unrelated.

9.27) (a), (b), (c), and (e)

The expressions ('c' + 'o' + 'o' + 'l') and ('o' + 'l') are of type int due to numeric promotion. Expression (d) is illegal since the String class has no constructor taking a single int parameter. Expression (a) is legal since string literals denote String objects and can be used just like any other object.

9.28) (d)
The constant expressions "ab" + "12" and "ab" + 12 will, at compile time, be evalutated to the string-valued constant "ab12". Both variables s and t are assigned a reference to the same interned String object containing "ab12". The variable u is assigned a new String object, created using the new operator.

9.29) (a), (c), and (d)
The String class does not have a constructor that takes a singlein t as a parameter.

9.30) (e)
The String class has no reverse() method.

9.31) (d)
The expression "abcdef".charAt(3) evaluates to the character 'd'. The charAt() method takes an int value as an argument and returns a char value. The expression ("abcdef").charAt(3) is legal. It also evaluates to the character 'd'. The index of the first character in a string is 0.

9.32) (e)
The expression "Hello there".toLowerCase().equals("hello there") will evaluate to true. The equals() method in the String class will only return true if the two strings have the same sequence of characters.

9.33) (c)
The variable middle is assigned the value 6. The variable nt is assigned the string "nt". The substring "nt" occurs three times in the string "Contentment!", starting at indexes 2, 5, and 9. The call s.lastIndexOf(nt, middle) returns the start index of the last occurrence of "nt", searching backwards from position 6.

9.34) (b)
The code will fail to compile since the expression (s == sb) is illegal. It compares references of two classes that are not related.

9.35) (e)
The program will compile without errors and will print have a when run. The contents of the string buffer are truncated down to 6 characters.

9.36) (a), (b), and (d)
The StringBuffer class does not have a constructor that takes an array ocf har as a parameter.

9.37) (a)
The StringBuffer class does not define a trim() method.

9.38) (d)
The program will construct an immutable String object containing "eenny" and a StringBuffer object containing " miny". The concat() method returns a reference value to a new immutable String object containing "eenny meeny", but the reference value is not stored. The append() method appends the string " mo" to the string buffer.

Collections Mock Questions

10.1) Which most closely matches a description of a Java Map?
(a) A vector of arrays for a 2D geographic representation.
(b) A class for containing unique array elements.
(c) A class for containing unique vector elements.
(d) An interface that ensures that implementing classes cannot contain duplicate keys.

10.2) How does the set collection deal with duplicate elements?
(a) An exception is thrown if you attempt to add an element with a duplicate value.
(b) The add method returns false if you attempt to add an element with a duplicate value.
(c) A set may contain elements that return duplicate values from a call to the equals method.
(d) Duplicate values will cause an error at compile time.

10.3) Which of the following most closely describes a bitset collection?
(a) A class that contains groups of unique sequences of bits.
(b) A method for flipping individual bits in instance of a primitive type.
(c) An array of boolean primitives that indicate zeros or ones.
(d) A collection for storing bits as on-off information, like a vector of bits.

10.4) What is the result of attempting to compile and run the following code?

```
public class Test1{
   public static void main(String[] args)
   {
   Integer int1 = new Integer(10);
   Vector vec1 = new Vector();
   LinkedList list = new LinkedList();
   vec1.add(int1);
   list.add(int1);
   if(vec1.equals(list)) System.out.println("equal");
   else  System.out.println("not equal");
    }
}
```

(a)The code will fail to compile.
(b)Runtime error due to incompatible object comparison
(c)Will run and print "equal".
(d)Will run and print "not equal".

10.5) What is the result of attempting to compile and run the following code?

```
public class Test {

   public static void main(String[] args){
    Integer a = new Integer(4);
    Integer b = new Integer(8);
    Integer c = new Integer(4);
    HashSet hs = new HashSet();
    hs.add(a);
    hs.add(b);
    hs.add(c);
    System.out.println(hs);
   }
}
```

(a)Will print [8, 4]
(b)Will print [4, 8, 4]
(c)Will print [8, 4, 4]

10.6) What is the result of attempting to compile and run the following code?

```
public class Test {

   public static void main(String[] args){
    Integer a = new Integer(4);
```

```
                    Integer b = new Integer(8);
                    Integer c = new Integer(4);
                    TreeSet hs = new TreeSet();
                    ts.add(a);
                    ts.add(b);
                    ts.add(c);
                    System.out.println(ts);
                   }
                }
```

(a)Will print [8, 4]
(b)Will print [4, 8, 4]
(c)Will print [8, 4, 4]
(d)Will print [4, 8]
(e)Will print [4, 4, 8]

10.7) What will this print out ?

```
                public class Test {

                   public static void main(String[] args){
                    Integer a = new Integer(8);
                    Integer b = new Integer(4);
                    Integer c = new Integer(4);
                    Vector vec = new Vector();
                    Iterator itr;
                    vec.add(a);
                    vec.add(b);
                    vec.add(c);
                    itr = vec.iterator();
                       while (itr.hasNext()) {
                        System.out.println("" + itr.next());
                       }
                   }
                }
```

(a) 8 , 4 and 4
(b) 4 , 4 and 8
(c) 8 and 4
(d) 4 and 8

10.8) Which of these statements are true?
(a)HashTable is a sub class of Dictionary
(b)ArrayList is a sub class of Vector
(c)LinkedList is a subclass of ArrayList
(d)Stack is a subclass of Vector

10.9) Which of these statements are true?
(a)LinkedList extends List
(b)AbstractSet extends Set
(c)HashSet extends AbstractSet
(d)WeakHashMap extends HashMap
(e)TreeSet extends AbstractSet

10.10) Which of these statements are true?
(a)A HashSet does not permit duplicates
(b)A Vector permits duplicates
(c)A TreeSet is an ordered Set
(d)A LinkedList is sorted in descending order
(e)A LinkedList is sorted in ascending order

10.11) True or False? A WeakHashMap is synchronized.
(a)True
(b)False

10.12) True or False? A Set rejects duplicates and is ordered
(a)True
(b)False

10.13) Select the true statements
(a)AbstractSet extends AbstractCollection
(b)AbstractList extends AbstractCollection
(c)HashSet extends AbstractSet
(d)Vector extends AbstractList
(e)AbstrctSequentialList extends AbstractList
(f)LinkedList extends AbstrctSequentialList

10.14) Which of these are core interfaces in the collections framework? Select the three correct answers.
a. Set
b. Bag
c. LinkedList
d. Collection
e. Map

10.15) Which of these implementations are provided by the java.util package? Select the two correct answers.
a. HashList
b. HashMap
c. ArraySet
d. ArrayMap
e. TreeMap

10.16) What is the name of the interface used to represent collections that maintain non-unique elements in order? Select the one correct answer.
a. Collection
b. Set
c. SortedSet
d. List
e. Sequence

10.17) Which statements are true about collections? Select the two correct answers.
a. Some operations on a collection may throw anU nsupportedOperationException.
b. Methods calling optional operations in a collection must either catch an UnsupportedOperationException or declare it in their throws clause.
c. A List can have duplicate elements.
d. An ArrayList can only accommodate a fixed number of elements.
e. The Collection interface contains a method named get.

10.18) Methods calling optional operations in a collection must either catch an UnsupportedOperationException or declare it in their throws clause. What will be the result of attempting to compile and run the following program?

```
import java.util.*;
public class Sets {
public static void main(String[] args) {
HashSet set1 = new HashSet();
addRange(set1, 1);
ArrayList list1 = new ArrayList();
addRange(list1, 2);
TreeSet set2 = new TreeSet();
addRange(set2, 3);
LinkedList list2 = new LinkedList();
addRange(list2, 5);
set1.removeAll(list1);
list1.addAll(set2);
list2.addAll(list1);
```

```
set1.removeAll(list2);
System.out.println(set1);
}
static void addRange(Collection col, int step) {
for (int i = step*2; i<=25; i+=step)
col.add(new Integer(i));
}
}
```

Select the one correct answer.

a. The program will fail to compile since operations are performed on incompatible collection implementations.
b. The program will fail to compile since the TreeSet denoted by set2 has not been given a Comparator to use when sorting its elements.
c. The program will compile without error, but will throw anU nsupportedOperationException when run.
d. The program will compile without error and will print all primes below 25 when run.
e. The program will compile without error and will print some other sequence of numbers when run.

10.19) Which of these methods are defined in the Collection interface? Select the three correct answers.

a. add(Object o)
b. retainAll(Collection c)
c. get(int index)
d. iterator()
e. indexOf(Object o)

10.20) What will be the output from the following program?

```
import java.util.*;
public class Iterate {
public static void main(String[] args) {
List l = new ArrayList();
l.add("A"); l.add("B"); l.add("C"); l.add("D"); l.add("E");
ListIterator i = l.listIterator();
i.next(); i.next(); i.next(); i.next();
i.remove();
i.previous(); i.previous();
i.remove();
System.out.println(l);
};
};
```

Select the one correct answer.

a. It will print [A, B, C, D, E].
b. It will print [A, C, E].
c. It will print [B, D, E].
d. It will print [A, B, D].
e. It will print [B, C, E].
f. It will print throw aN oSuchElementException.

10.21) Which of these methods from the Collection interface will return the value true if the collection was modified during the operation? Select the two correct answers.

a. contains()
b. add()
c. containsAll()
d. retainAll()
e. clear()

10.22) Which of these methods can be called on objects implementing the Map interface? Select the two correct answers.

a. contains(Object o)
b. addAll(Collection c)
c. remove(Object o)
d. values()

e. toArray()

10.23) Which statements are true about maps? Select the two correct answers.
a. The return type of the values() method is Set.
b. Changes made in the set view returned by keySet() will be reflected in the original map.
c. The Map interface extends the Collection interface.
d. All keys in a map are unique.
e. All Map implementations keep the keys sorted.

10.24) Which sequence of digits will the following program print?

```
import java.util.*;
public class Lists {
public static void main(String[] args) {
List list = new ArrayList();
list.add("1");
list.add("2");
list.add(1, "3");
List list2 = new LinkedList(list);
list.addAll(list2);
list2 = list.subList(2, 5);
list2.clear();
System.out.println(list);
}
}
```

Select the one correct answer.
a. [1, 3, 2]
b. [1, 3, 3, 2]
c. [1, 3, 2, 1, 3, 2]
d. [3, 1, 2]
e. [3, 1, 1, 2]
f. None of the above.

10.25) Which of these classes have a comparator() method? Select the two correct answers.
a. ArrayList
b. HashMap
c. TreeSet
d. HashSet
e. TreeMap

10.26) Which method prototypes are defined in the interface java.util.Map.Entry? Select the two correct answers.
a. Object getKey()
b. Object setKey(Object value)
c. void remove()
d. Object getValue()
e. void setValue(Object value)

10.27) Given that the objects denoted by the parameters override the equals() and the hashCode() methods appropriately, which return values are possible from the following method?

```
String func(Object x, Object y) {
return (x == y) + " " + x.equals(y) + " " + (x.hashCode() ==
y.hashCode());
}
```

Select the two correct answers.
a. "false false true"
b. "false true false"
c. "false true true"
d. "true false false"
e. "true false true"

10.28) Insert code into the equalsImpl() method in order to provide a correct implementation of the equals() method.

```
public class Pair {
int a, b;
public Pair(int a, int b) {
this.a = a;
this.b = b;
}
public boolean equals(Object o) {
return (this == o) || (o instanceof Pair) && equalsImpl((Pair) o);
}
private boolean equalsImpl(Pair o) {
// ... PROVIDE IMPLEMENTATION HERE ...
}
}
```

Select the three correct answers.

a. return a == o.a || b == o.b;
b. return false;
c. return a >= o.a;
d. return a == o.a;
e. return a == o.a && b == o.b;

10.29) Which collection implementation is thread-safe? Select the one correct answer.

a. ArrayList
b. HashSet
c. Vector
d. TreeSet
e. LinkedList

10.30) Which code provides a correct implementation of the hashCode() method in the following program?

```
import java.util.*;
public class Measurement {
int count;
int accumulated;
public Measurement() {}
public void record(int v) {
count++;
accumulated += v;
}
public int average() {
return accumulated/count;
}
public boolean equals(Object other) {
if (this == other)
return true;
if (!(other instanceof Measurement))
return false;
Measurement o = (Measurement) other;
if (count != 0 && o.count != 0)
return average() == o.average();
return count == o.count;
}
public int hashCode() {
// ... PROVIDE IMPLEMENTATION HERE ...
}
}
```

Select the two correct answers.

a. return 31337;
b. return accumulated / count;
c. return (count << 16) ^ accumulated;
d. return ~accumulated;
e. return count == 0 ? 0 : average();

Collections Answers

10.1) (d)

10.2) (b)

10.3) (d)
This is the description given to a bitset in Bruce Eckels "Thinking in Java" book. The reference to unique sequence of bits was an attempt to mislead because of the use of the word Set in the name bitset. Normally something called a set implies uniqueness of the members, but not in this context.

10.4) (c)

10.5) (a)

10.6) (d)

10.7) (a)

10.8) (a), (d)

10.9) (c), (e)

10.10) (a), (b), (c)

10.11) (b)

10.12) (b)

10.13) (a), (b), (c), (d), (e), (f)

10.14) (a), (d), and (e)
Set, Collection and Map are core interfaces in the collections framework. LinkedList is a class that implements the List interface. There is no class or interface named Bag.

10.15) (b) and (e)
The java.util package provides map implementations named HashMap and TreeMap. It does not provide any implementations named HashList, ArraySet, and ArrayMap.

10.16) (d)
The List interface is implemented by collections that maintain sequences of possibly non-unique elements. Elements retain their ordering in the sequence. Collection classes implementing SortedSet only allow unique elements that are maintained in a sorted order.

10.17) (a) and (c)
Some operations on a collection may throw an UnsupportedOperationException. This exception type is unchecked, and the code is not required to explicitly handle unchecked exceptions. A List allows duplicate elements. An ArrayList implements a resizable array. The capacity of the array will be expanded automatically when needed. The List interface defines a get() method, but there is no method by that name in the Collection interface.

10.18) (d)
The program will compile without error, and will print all primes below 25 when run. All the collection implementations used in the program implement the Collection interface. The implementation instances are interchangeable when denoted by Collection references. None of the operations performed on the implementations will throw an UnsupportedOperationException. The program finds the primes below 25 by removing all values divisible by 2, 3, and 5 from the set of values from 2 through 25.

10.19) (a), (b), and (d)

The methods add(), retainAll(), and iterator() are defined in the Collection interface. The get() and indexOf() methods are defined in the List interface.

10.20) (b)
The remove() method removes the last element returned by either next() or previous(). The four next() calls return A, B, C, and D. D is subsequently removed. The two previous() calls return C and B. B is subsequently removed.

10.21) (b) and (d)
The methods add() and retainAll(), return the value true if the collection was modified during the operation. The contains() and containsAll() methods return a boolean value, but these membership operations never modify the current collection, and the return value indicates the result of the membership test. The clear() method does not have a return value.

10.22) (c) and (d)
The Map interface defines the methods remove() and values(). It does not define methods contains(), addAll(), and toArray(). Methods with these names are defined in the Collection interface, but Map does not inherit from Collection.

10.23) (b) and (d)
Although all the keys in a map must be unique, multiple identical values may exist. Since values are not unique, the values() method returns a Collection instance and not a Set instance. The collection objects returned by the keySet(), entrySet(), and values() methods are backed by the original Map object. This means that changes made in one are reflected in the other. Although implementations of SortedMap keep the entries sorted on the keys, this is not a requirement for classes that implement Map. For instance, the entries in a HashMap are not sorted.

10.24) (a)
[1, 3, 2] is printed. First, "1" and "2" are appended to an empty list. Next, "3" is inserted between "1" and "2", and then the list is duplicated. The original list is concatenated with the copy. The sequence of elements in the list is now "1", "3", "2", "1", "3", "2". Then a sublist view allowing access to elements from index 2 to index 5 (exclusive) is created (i.e., the subsequence "2", "1", "3"). The sublist is cleared, thus removing the elements. This is reflected in the original list and the sequence of elements is now "1", "3", "2".

10.25) (c) and (e)
The classes TreeSet and TreeMap implement the comparator() method. The comparator() method is defined in the SortedSet and SortedMap interfaces, and the TreeSet and TreeMap classes implement these interfaces.

10.26) (a) and (d)
The key of a Map.Entry cannot be changed since the key is used for locating the entry within the list. Although iterators obtained for the entry set of a map have a remove() method, the entries themselves do not. Map.Entry has a method named setValue, but its return type is Object.

10.27) (a) and (c)
(b) is eliminated since the hashCode() method cannot claim inequality if the equals() method claims equality. (d) and (e) are eliminated since the equal() method must be reflexive, and the hashCode() method must consistently return the same hash value during the execution.

10.28) (b), (d), and (e)
(a) and (c) fail to satisfy the properties of an equivalence relation. (a) is not transitive and (c) is not symmetric.

10.29) (c)
Of all the collection classes in the java.util package, only Vector and HashTable are thread-safe. The Collections class contains a static synchronizedCollectionType() method that creates thread-safe instances based on collections, which are not.

10.30) (a) and (e)

(b) is not correct since it will throw an ArithmeticException when called on a newly created Measurement object. (c) and (d) are not correct since they may return unequal hash values for two objects that are equal according to the equals() method.

AWT Mock Questions

11.1) What will be the result of compiling and running the following code?

```
import java.awt.*;
import java.applet.*;
public class AppletTest extends Applet {
  Label lbl = new Label("hello");
   public void init()
   {
   setSize(200,100);
   setVisible(true);
   lbl.setBackground(new Color(0,100,180));
   setLayout(new GridLayout(1,1));
   add(lbl);
   setLayout(new FlowLayout());
   lbl.setBounds(0,0,100,24);
   }
}
```

(a) The label will fill half the display area of the applet.
(b) The label will be wide enough to display the text "hello"
(c) The label will not be visiblle.
(d) The label will fill the entire display area of the applet
(e) The code will throw a run time error because of the second setLayout() call.

11.2) What will be the result of compiling and running the following applet?

```
import java.awt.*;
import java.applet.*;
public class AppletTest extends Applet {
   public void init()
   {
   super.init();
   PanelTest p = new PanelTest();
   p.init();
   setVisible(true);
   setSize(200,100);
   add(p);
}

class PanelTest extends Panel{
   Button b1 = new Button("Press me");
   public PanelTest()
   {
   setSize(200,100);
   setVisible(true);
   }
   public void init(){
   super.init();
   add(b1);
   }
  }
}
```

(a) The button will fill the entire display area of the applet.
(b) The code will fail to compile.
(c) The button will be just big enough to encompass it's label.
(d) The applet's display area will be blank.

11.3) This code compiles without error.

```
class MyButton extends Button implements MouseListener{
 public MyButton(String lbl) {
   super(lbl);
   addMouseListener(this);
 }
```

```
 public void mousePressed(MouseEvent e){
 //do something
  }
}
```

(a) False
(b) True

11.4) Which of the following are valid constructors for a TextField .
(a) TextField();
(b) TextField(int rows , int cols);
(c) TextField(int cols , String txt);
(d) TextField(int cols);
(e) TextField(String txt , boolean scrollBars);

11.5) True or false? void setResizable(boolean) is a member of the Applet class.
(a) True
(b) False

11.6) What is the result of attempting to compile and run the following.

```
import java.awt.*;
import java.awt.event.*;
class FrameTest extends Frame{
   Label lblTest  = new Label("TEST");
   Button btnTest = new Button(" TEST ");
   public static void main(String[] args){
     FrameTest ft = new FrameTest();
     ft.setLayout(new FlowLayout());
     ft.add(ft.lblTest);
     ft.add(ft.btnTest);
     ft.setSize(200,100);
     ft.setVisible(true);
     ft.enableEvents(AWTEvent.ACTION_EVENT_MASK);
    }
   public void processActionEvent(ActionEvent event){
      super.processActionEvent(event);
        if(event.getID() == AWTEvent.ACTION_EVENT_MASK){
          if(event.getSource() instanceof Button){
           lblTest.setText("OK");
           }
        }
    }
}
```

(a) The code will not compile
(b) There will be a runtime error
(c) The frame wil not be visible
(d) Nothing happens when you click the button
(e) The label's caption changes to "OK" when you click the button

11.7) What is the default layout for a Dialog ?
(a) FlowLayout
(b) GridLayout
(c) CardLayout
(d) BorderLayout
(e) GridBagLayout

11.8) True or false? A Dialog is a subclass of Frame.
(a) True
(b) False

11.9) Which of the following are valid constructors for a MenuItem ?
(a) MenuItem()

(b) MenuItem(String name)
(c) MenuItem(String name , boolean removable)
(d) MenuItem(String name , MenuShortcut sc)
(e) MenuItem(boolean check)

11.10)What will this draw ?

```
public class AppletTest extends Applet{
public void init(){
setVisible(true);
setSize(200,200);
}
public void paint(Graphics g){
g.setColor(new Color(0,0,255));
g.drawRect(50, 100 , 100, 50);
}
}
```

(a) A rectangle 50 pixels wide with top left corner at 100, 50.
(b) A rectangle 50 pixels wide with top left corner at 50,100.
(c) A rectangle 100 pixels wide with top left corner at 50,100.
(d) A rectangle 100 pixels high with top left corner at 50,100.
(e) A rectangle 100 pixels high with top left corner at 100,50.

11.11) What most closely matches the appearance when this code runs? [7]

```
import java.awt.*;
public class CompLay extends Frame{
public static void main(String argv[]){
    CompLay cl = new CompLay();
    }

CompLay(){
    Panel p = new Panel();
    p.setBackground(Color.pink);
    p.add(new Button("One"));
    p.add(new Button("Two"));
    p.add(new Button("Three"));
    add("South",p);
    setLayout(new FlowLayout());
    setSize(300,300);
    setVisible(true);
    }
}
```

(a) The buttons will run from left to right along the bottom of the Frame.
(b) The buttons will run from left to right along the top of the frame.
(c) The buttons will not be displayed.
(d) Only button three will show occupying all of the frame.

11.12) What will happen when you attempt to compile and run this code? [7]

```
//Demonstration of event handling

import java.awt.event.*;
import java.awt.*;
public class MyWc extends Frame implements WindowListener{
public static void main(String argv[]){
        MyWc mwc = new MyWc();
        }
        public void windowClosing(WindowEvent we){
                System.exit(0);
                }//End of windowClosing
      public void  MyWc(){
        setSize(300,300);
        setVisible(true);
        }
```

```
}//End of class
```

(a) Error at compile time.
(b) Visible Frame created that that can be closed.
(c) Compilation but no output at run time.
(d) Error at compile time because of comment before *import* statements.

AWT Answers

11.1) (b)

11.2) (b)

11.3) (a)

11.4) (a), (d)

11.5) (b)

11.6) (a)

11.7) (d)

11.8) (b)

11.9) (a), (b), (d)

11.10) (c)

11.11) (b)
The call to the setLayout(new FlowLayout()) resets the Layout manager for the entire frame and so the buttons end up at the top rather than the bottom.

11.12) (a)
If you implement an interface you must create bodies for all methods in that interface. This code will produce an error saying that MyWc must be declared abstract because it does not define all of the methods in WindowListener. (d) is nonsense as comments can appear anywhere. (c) suggesting that it might compile but not produce output is meant to mislead on the basis that what looks like a constructor is actually an ordinary method as it has a return type.

Files & Streams Mock Questions

12.1) Which of the following can you perform using the File class? [4]
(a) Change the current directory.
(b) Return the name of the parent directory.
(c) Delete a file.
(d) Find if a file contains text or binary information.

12.2) You can create a RandomAccessFile object in the following manner
```
FileInputStream fis = new FileInputStream("file path");
RandomAccessFile raf = new RandomAccessFile(fis , "rw");
```
(a)False.
(b)True

12.3) What is the result of attempting to compile and run this ?
```
public class Test {
     public static void main(String[] args){
     byte out = 0;
     RandomAccessFile raf;
          try{
          raf = new RandomAccessFile("test1.txt" , "rw");
           for (int i=0; i<10; i++)
          raf.write(i);
          raf.seek(1);
          out = raf.readByte();
          raf.close();
          System.out.println(out);
          }
          catch(IOException e){System.out.println("err");}
      }
    }
```
(a)1
(b)0

12.4) What is the result of attempting to compile and run this ?
```
public class Test {
  public static void main(String[] args){
  byte out = 0;
  String  f = "test.txt";
  RandomAccessFile raf;
        try{
        raf = new RandomAccessFile(f , "rw");
        raf.write(1);
        raf.close();
        raf.open(f, true);
        raf.seek(1);
        out = raf.readByte();
        raf.close();
        System.out.println(out);
        }
        catch(IOException e){System.out.println("err");}
   }
  }
```
(a)It will run and print "err"
(b)It will run and print "1"
(c)It will throw a runtime exception
(d)It will not compile

12.5) Which of the fllowing will create a file called test.txt in your current directory ?
(a)File f = new File("test.txt");

(b)FileOutputStream fos = new FileOutputStream("test.txt");
(c)RandomAccessFile raf = new RandomAccessFile("test.txt", "r");
(d)RandomAccessFile raf = new RandomAccessFile("test.txt", "rw");

12.6) True or false? UTF characters are always 16 bits
(a)True
(b)False

12.7)

```
void aMethod(){
     File f = new File("test.txt");
   try{ f.createNewFile();}
      catch(Exception e){System.out.println("err");}
  }
```

Calling this method will overwrite the existing file an create a blank file in its place.
(a)True
(b)False

12.8) Which of the following are valid constructors for a DataInputStream ?
(a)DataInputStream(FileInputStream fis)
(b)DataInputStream(DataInputStream dis)
(c)DataInputStream(File f)
(d)DataInputStream(FilterInputStream fis)

12.9) True or false ? InputStreamReader is a subclass of InputStream
(a)True
(b)False

12.10) This is a valid constructor for a BufferedReader.

```
BufferedReader(FileReader fr)
```

True or false?
(a)True
(b)False

Files & Streams Answers

12.1) (b), (c)
It is surprising that you can't change the current directory. It is not so surprising that you can't tell if a file contains text or binary information.

12.2) (a)

12.3) (a)

12.4) (d)

12.5) (b), (d)

12.6) (b)

12.7) (b)

12.8) (a), (b), (d)

12.9) (b)

12.10) (a)

Mock Exam

This is a mock exam. It comprises of questions, which are similar to the questions that can be expected on the real exam. Working through this exam will give the reader a good indication of how well he/she is prepared for the real exam, and whether any topics need further study.

Q1 Given the following class, which statements can be inserted at position 1 without causing a compilation error?

```
public class Q6db8 {
int a;
int b = 0;
static int c;
public void m() {
int d;
int e = 0;
// Position 1
}
}
```

Select the four correct answers.

a. a++;
b. b++;
c. c++;
d. d++;
e. e++;

Q2 Which statements are true about the effect of the >> and >>> operators? Select the three correct answers.

a. For non-negative values of the left operand, the> > and >>> operators will have the same effect.
b. The result of (-1 >> 1) is 0.
c. The result of (-1 >>> 1) is –1.
d. The value returned by >>> will never be negative if the left operand is non-negative.
e. When using the >> operator, the left-most bit of the resulting value will always have the same bit value as the left-most bit of the left operand.

Q3 What is wrong with the following code?

```
class MyException extends Exception {}
public class Qb4ab {
public void foo() {
try {
bar();
} finally {
baz();
} catch (MyException e) {}
}
public void bar() throws MyException {
throw new MyException();
}
public void baz() throws RuntimeException {
throw new RuntimeException();
}
}
```

Select the one correct answer.

a. Since the method foo() does not catch the exception generated by the method baz(), it must declare the RuntimeException in a throws clause.
b. A try block cannot be followed by both ac atch and a finally block.
c. An empty catch block is not allowed.
d. A catch block cannot follow a finally block.
e. A finally block must always follow one or morec atch blocks.

Q4 What will be written to the standard output when the following program is run?

```
public class Qd803 {
public static void main(String[] args) {
String word = "restructure";
System.out.println(word.substring(2, 3));
}
}
```

Select the one correct answer.

a. est
b. es
c. str
d. st
e. s

Q5 Given that a static method doIt() in the class Work represents work to be done, which block of code will succeed in starting a new thread that will do the work? Select the one correct answer.

a.

```
Runnable r = new Runnable() {
public void run() {
Work.doIt();
}
};
Thread t = new Thread(r);
t.start();
```

b.

```
Thread t = new Thread() {
public void start() {
Work.doIt();
}
};
t.start();
```

c.

```
Runnable r = new Runnable() {
public void run() {
Work.doIt();
}
};
r.start();
```

d.

```
Thread t = new Thread(new Work());
t.start();
```

e.

```
Runnable t = new Runnable() {
public void run() {
Work.doIt();
}
};
t.run();
```

Q6 What will be printed when the following program is run?

```
public class Q8929 {
public static void main(String[] args) {
for (int i=12; i>0; i-=3)
System.out.print(i);
System.out.println("");
}
}
```

Select the one correct answer.

a. 12
b. 129630
c. 12963

d. 36912
e. None of the above.

Q7 What will be the result of attempting to compile and run the following code?

```
public class Q275d {
static int a;
int b;
public Q275d() {
int c;
c = a;
a++;
b += c;
}
public static void main(String[] args) {
new Q275d();
}
}
```

Select the one correct answer.

a. The code will fail to compile since the constructor is trying to access static members.
b. The code will fail to compile since the constructor is trying to use static fielda before it has been initialized.
c. The code will fail to compile since the constructor is trying to use fieldb before it has been initialized.
d. The code will fail to compile since the constructor is trying to use local variable c before it has been initialized.
e. The code will compile and run without any problems.

Q8 What will be written to the standard output when the following program is run?

```
public class Q63e3 {
public static void main(String[] args) {
System.out.println(9 ^ 2);
}
}
```

Select the one correct answer.

a. 81
b. 7
c. 11
d. 0
e. false

Q9 Which statements is true about the compilation and execution of the following program with assertions enabled?

```
public class Qf1e3 {
String s1;
String s2 = "hello";
String s3;
Qf1e3() {
s1 = "hello";
}
public static void main(String[] args) {
(new Qf1e3()).f();
}
{
s3 = "hello";
}
void f() {
String s4 = "hello";
String s5 = new String("hello");
assert(s1.equals(s2)); // (1)
assert(s2.equals(s3)); // (2)
assert(s3 == s4); // (3)
assert(s4 == s5); // (4)
```

```
}
}
```

Select the one correct answer.

a. The compilation will fail.
b. The assertion on the line marked (1) will fail.
c. The assertion on the line marked (2) will fail.
d. The assertion on the line marked (3) will fail.
e. The assertion on the line marked (4) will fail.
f. The program will run without any errors.

Q10 Which declarations of the main() method are valid in order to start the execution of an application? Select the two correct answers.

a. public void main(String args[])
b. public void static main(String args[])
c. public static main(String[] argv)
d. final public static void main(String [] array)
e. public static void main(String args[])

Q11 Under which circumstance will a thread stop? Select the one correct answer.

a. The run() method that the thread is executing ends.
b. The call to the start() method of the Thread object returns.
c. The suspend() method is called on the Thread object.
d. The wait() method is called on the Thread object.

Q12 When creating a class that associates a set of keys with a set of values, which of these interfaces is most applicable? Select the one correct answer.

a. Collection
b. Set
c. SortedSet
d. Map

Q13 What is the result of running the following code with assertions enabled?

```
public class Q1eec {
static void test(int i) {
int j = i/2;
int k = i >>> 1;
assert j == k : i;
}
public static void main(String[] args) {
test(0);
test(2);
test(-2);
test(1001);
test(-1001);
}
}
```

Select the one correct answer.

a. The program executes normally and produces no output.
b. An AssertionError with 0 as message is thrown.
c. An AssertionError with 2 as message is thrown.
d. An AssertionError with -2 as message is thrown.
e. An AssertionError with 1001 as message is thrown.
f. An AssertionError with -1001 as message is thrown.

Q14 What will be written to the standard output when the following program is run?

```
class Base {
int i;
Base() { add(1); }
void add(int v) { i += v; }
void print() { System.out.println(i); }
}
```

```
class Extension extends Base {
Extension() { add(2); }
void add(int v) { i += v*2; }
}
public class Qd073 {
public static void main(String[] args) {
bogo(new Extension());
}
static void bogo(Base b) {
b.add(8);
b.print();
}
}
```

Select the one correct answer.

a. 9
b. 18
c. 20
d. 21
e. 22

Q15 Which declarations of a native method are valid in the declaration of the following class?

```
public class Qf575 {
// Insert declaration of a native method here
}
```

Select the two correct answers.

a. native public void setTemperature(int kelvin);
b. private native void setTemperature(int kelvin);
c. protected int native getTemperature();
d. public abstract native void setTemperature(int kelvin);
e. native int setTemperature(int kelvin) {}

Q16 Which collection implementation is suitable for maintaining an ordered sequence of objects, when objects are frequently inserted and removed from the middle of the sequence? Select the one correct answer.

a. TreeMap
b. HashSet
c. Vector
d. LinkedList
e. ArrayList

Q17 Which statements can be inserted at the indicated position in the following code to make the program print 1 on the standard output when executed?

```
public class Q4a39 {
int a = 1;
int b = 1;
int c = 1;
class Inner {
int a = 2;
int get() {
int c = 3;
// Insert statement here.
return c;
}
}
Q4a39() {
Inner i = new Inner();
System.out.println(i.get());
}
public static void main(String[] args) {
new Q4a39();
}
}
```

Select the two correct answers.

a. c = b;
b. c = this.a;
c. c = this.b;
d. c = Q4a39.this.a;
e. c = c;

Q18 Which is the earliest line in the following code after which the object created in the line marked (0) will be a candidate for garbage collection, assuming no compiler optimizations are done?

```
public class Q76a9 {
static String f() {
String a = "hello";
String b = "bye"; // (0)
String c = b + "!"; // (1)
String d = b; // (2)
b = a; // (3)
d = a; // (4)
return c; // (5)
}
public static void main(String[] args) {
String msg = f();
System.out.println(msg); // (6)
}
}
```

Select the one correct answer.

a. The line marked (1).
b. The line marked (2).
c. The line marked (3).
d. The line marked (4).
e. The line marked (5).
f. The line marked (6).

Q19 Which method from the String and StringBuffer classes modifies the object on which it is invoked?
Select the one correct answer.

a. The charAt() method of the String class.
b. The toUpperCase() method of the String class.
c. The replace() method of the String class.
d. The reverse() method of the StringBuffer class.
e. The length() method of the StringBuffer class.

Q20 Which statement, when inserted at the indicated position in the following code, will cause a runtime exception?

```
class A {}
class B extends A {}
class C extends A {}
public class Q3ae4 {
public static void main(String[] args) {
A x = new A();
B y = new B();
C z = new C();
// Insert statement here
}
}
```

Select the one correct answer.

a. x = y;
b. z = x;
c. y = (B) x;
d. z = (C) y;
e. y = (A) y;

Q21 Which of these are keywords in Java? Select three correct answers.

a. default
b. NULL
c. String
d. throws
e. long

Q22 A method within a class is only accessible by classes that are defined within the same package as the class of the method. How can such a restriction be enforced? Select the one correct answer.

a. Declare the method with the keyword public.
b. Declare the method with the keyword protected.
c. Declare the method with the keyword private.
d. Declare the method with the keyword package.
e. Do not declare the method with any accessibility modifiers.

Q23 Which code initializes the two-dimensional array tab so that tab[3][2] is a valid element? Select the two correct answers.

a.
```
int[][] tab = {
{ 0, 0, 0 },
{ 0, 0, 0 }
};
```
b.
```
int tab[][] = new int[4][];
for (int i=0; i<tab.length; i++) tab[i] = new int[3];
```
c.
```
int tab[][] = {
0, 0, 0, 0,
0, 0, 0, 0,
0, 0, 0, 0,
0, 0, 0, 0
};
```
d. int tab[3][2];
e. int[] tab[] = { {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0} };

Q24 What will be the result of attempting to run the following program?

```
public class Qaa75 {
public static void main(String[] args) {
String[][][] arr = {
{ {}, null },
{ { "1", "2" }, { "1", null, "3" } },
{},
{ { "1", null } }
};
System.out.println(arr.length + arr[1][2].length);
}
}
```

Select the one correct answer.

a. The program will terminate with an ArrayIndexOutOfBoundsException.
b. The program will terminate with a NullPointerException.
c. 4 will be written to standard output.
d. 6 will be written to standard output.
e. 7 will be written to standard output.

Q25 Which expressions will evaluate to true if preceded by the following code?

```
String a = "hello";
String b = new String(a);
String c = a;
char[] d = { 'h', 'e', 'l', 'l', 'o' };
```

Select the two correct answers.

a. (a == "Hello")
b. (a == b)

c. (a == c)
d. a.equals(b)
e. a.equals(d)

Q26 Which statements are true about the following code?

```
class A {
public A() {}
public A(int i) { this(); }
}
class B extends A {
public boolean B(String msg) { return false; }
}
class C extends B {
private C() { super(); }
public C(String msg) { this(); }
public C(int i) {}
}
```

Select the two correct answers.

a. The code will fail to compile.
b. The constructor in A that takes an int as an argument will never be called as a result of constructing an object of class B or C.
c. Class C defines three constructors.
d. Objects of class B cannot be constructed.
e. At most one of the constructors of each class is called as a result of constructing an object of clasCs .

Q27 Given two collection objects referenced by col1 and col2, which statements are true? Select the two correct answers.

a. The operation col1.retainAll(col2) will not modify the col1 object.
b. The operation col1.removeAll(col2) will not modify the col2 object.
c. The operation col1.addAll(col2) will return a new collection object, containing elements from bothc ol1 and col2.
d. The operation col1.containsAll(Col2) will not modify the col1 object.

Q28 Which statements are true about the relationships between the following classes?

```
class Foo {
int num;
Baz comp = new Baz();
}
class Bar {
boolean flag;
}
class Baz extends Foo {
Bar thing = new Bar();
double limit;
}
```

Select the three correct answers.

a. A Bar is a Baz.
b. A Foo has a Bar.
c. A Baz is a Foo.
d. A Foo is a Baz.
e. A Baz has a Bar.

Q29 Which statements are true about the value of a field, when no explicit assignments have been made? Select the three correct answers.

a. The value of a field of type int is undetermined.
b. The value of a field of any numeric type is zero.
c. The compiler may issue an error if the field is used in a method before it is initialized.
d. A field of type String will denote the empty string ("").
e. The value of all fields which are references is null.

Q30 Which statements describe guaranteed behavior of the garbage collection and finalization mechanisms? Select the two correct answers.

a. An object is deleted as soon as there are no more references that denote the object.
b. The finalize() method will eventually be called on every object.
c. The finalize() method will never be called more than once on an object.
d. An object will not be garbage collected as long as it is possible for a live thread to access it through a reference.
e. The garbage collector will use a mark and sweep algorithm.

Q31 Which main() method will succeed in printing the last program argument to the standard output, and exit gracefully with no output if no program arguments are specified? Select the one correct answer.

a.
```
public static void main(String[] args) {
if (args.length != 0)
System.out.println(args[args.length-1]);
}
```
b.
```
public static void main(String[] args) {
try { System.out.println(args[args.length]); }
catch (ArrayIndexOutOfBoundsException e) {}
}
```
c.
```
public static void main(String[] args) {
int ix = args.length;
String last = args[ix];
if (ix != 0) System.out.println(last);
}
```
d.
```
public static void main(String[] args) {
int ix = args.length-1;
if (ix > 0) System.out.println(args[ix]);
}
```
e.
```
public static void main(String[] args) {
try { System.out.println(args[args.length-1]); }
catch (NullPointerException e) {}
}
```

Q32 Which statements are true about the collection interfaces? Select the three correct answers.

a. Set extends Collection.
b. All methods defined in Set are also defined in Collection.
c. List extends Collection.
d. All methods defined in List are also defined in Collection.
e. Map extends Collection.

Q33 Which is the legal range of values for a short? Select the one correct answer.

a. -2^7 to 2^7-1
b. -2^8 to 2^8
c. -2^{15} to $2^{15}-1$
d. -2^{16} to $2^{16}-1$
e. 0 to $2^{16}-1$

Q34 What is the name of the method that threads can use to pause their execution until signalled to continue by another thread? Fill in the name of the method (do not include a parameter list).

Q35 Given the following class definitions, which expression identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C, and D?

```
class A {}
class B extends A {}
class C extends B {}
class D extends A {}
```

Select the one correct answer.

a. obj instanceof B
b. obj instanceof A && !(obj instanceof C)
c. obj instanceof B && !(obj instanceof C)
d. !(obj instanceof C || obj instanceof D)
e. !(obj instanceof A) && !(obj instanceof C) && !(obj instanceof D)

Q36 What will be written to the standard output when the following program is executed?

```
public class Q8499 {
public static void main(String[] args) {
double d = -2.9;
int i = (int) d;
i *= (int) Math.ceil(d);
i *= (int) Math.abs(d);
System.out.println(i);
}
}
```

Select the one correct answer.

a. -12
b. 18
c. 8
d. 12
e. 27

Q37 What will be written to the standard output when the following program is executed?

```
public class Qcb90 {
int a;
int b;
public void f() {
a = 0;
b = 0;
int[] c = { 0 };
g(b, c);
System.out.println(a + " " + b + " " + c[0] + " ");
}
public void g(int b, int[] c) {
a = 1;
b = 1;
c[0] = 1;
}
public static void main(String[] args) {
Qcb90 obj = new Qcb90();
obj.f();
}
}
```

Select the one correct answer.

a. 0 0 0
b. 0 0 1
c. 0 1 0
d. 1 0 0
e. 1 0 1

Q38 Given the following class, which are correct implementations of the hashCode() method?

```
class ValuePair {
public int a, b;
public boolean equals(Object other) {
```

```
try {
ValuePair o = (ValuePair) other;
return (a == o.a && b == o.b)
|| (a == o.b && b == o.a);
} catch (ClassCastException cce) {
return false;
}
}
public int hashCode() {
// Provide implementation here.
}
}
```

Select the three correct answers.

a. return 0;
b. return a;
c. return a + b;
d. return a - b;
e. return a ^ b;
f. return (a << 16) | b;

Q39 Which statements are true regarding the execution of the following code?

```
public class Q3a0a {
public static void main(String[] args) {
int j = 5;
for (int i = 0; i<j; i++) {
assert i < j-- : i > 0;
System.out.println(i*j);
}
}
}
```

Select the two correct answers.

a. An AssertionError will be thrown if assertions are enabled at runtime.
b. The last number printed is 4 if assertions are disabled at runtime.
c. The last number printed is 20 if assertions are disabled at runtime.
d. The last number printed is 4 if assertions are enabled at runtime.
e. The last number printed is 20 if assertions are enabled at runtime.

Q40 Which of the following method names are overloaded? Select the three correct answers.

a. The method name yield in java.lang.Thread
b. The method name sleep in java.lang.Thread
c. The method name wait in java.lang.Object
d. The method name notify in java.lang.Object

Q41 Which are valid identifiers? Select the three correct answers.

a. _class
b. $value$
c. zer@
d. ångström
e. 2much

Q42 What will be the result of attempting to compile and run the following program?

```
public class Q28fd {
public static void main(String[] args) {
int counter = 0;
l1:
for (int i=0; i<10; i++) {
l2:
int j = 0;
while (j++ < 10) {
if (j > i) break l2;
if (j == i) {
counter++;
```

```
continue l1;
}
}
}
System.out.println(counter);
}
}
```

Select the one correct answer.

a. The program will fail to compile.
b. The program will not terminate normally.
c. The program will write 10 to the standard output.
d. The program will write 0 to the standard output.
e. The program will write 9 to the standard output.

Q43 Given the following interface definition, which definition is valid?

```
interface I {
void setValue(int val);
int getValue();
}
```

Select the one correct answer.

a.
```
class A extends I {
int value;
void setValue(int val) { value = val; }
int getValue() { return value; }
}
```
b.
```
interface B extends I {
void increment();
}
```
c.
```
abstract class C implements I {
int getValue() { return 0; }
abstract void increment();
}
```
d.
```
interface D implements I {
void increment();
}
```
e.
```
class E implements I {
int value;
public void setValue(int val) { value = val; }
}
```

Q44 Which statements are true about the methods notify() and notifyAll()? Select the two correct answers.

a. An instance of the class Thread has a method named notify that can be invoked.
b. A call to the method notify() will wake the thread that currently owns the lock of the object.
c. The method notify() is synchronized.
d. The method notifyAll() is defined in class Thread.
e. When there is more than one thread waiting to obtain the lock of an object, there is no way to be sure which thread will be notified by the notify() method.

Q45 Which statements are true about the correlation between the inner and outer instances of member classes? Select the two correct answers.

a. Fields of the outer instance are always accessible to inner instances, regardless of their accessibility modifiers.
b. Fields of the outer instance can never be accessed using only the variable name within the inner instance.
c. More than one inner instance can be associated with the same outer instance.

d. All variables from the outer instance that should be accessible in the inner instance must be declaredfi nal.
e. A class that is declared final cannot have any member classes.

Q46 What will be the result of attempting to compile and run the following code?

```
public class Q6b0c {
public static void main(String[] args) {
int i = 4;
float f = 4.3;
double d = 1.8;
int c = 0;
if (i == f) c++;
if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
System.out.println(c);
}
}
```

Select the one correct answer.
a. The code will fail to compile.
b. The value 0 will be written to the standard output.
c. The value 1 will be written to the standard output.
d. The value 2 will be written to the standard output.
e. The value 3 will be written to the standard output.

Q47 Which operators will always evaluate all the operands? Select the two correct answers.
a. ||
b. +
c. &&
d. ? :
e. %

Q48 Which statement concerning the switch construct is true? Select the one correct answer.
a. All switch statements must have ad efault label.
b. There must be exactly one label for each code segment in as witch statement.
c. The keyword continue can never occur within the body of as witch statement.
d. No case label may follow a default label within a single switch statement.
e. A character literal can be used as a value for ac ase label.

Q49 Which modifiers and return types would be valid in the declaration of a main() method that starts the execution of a Java standalone application? Select the two correct answers.
a. private
b. final
c. static
d. int
e. abstract
f. String

Q50 Which of the following expressions are valid? Select the three correct answers.
a. System.out.hashCode()
b. "".hashCode()
c. 42.hashCode()
d. ("4"+2).equals(42)
e. (new java.util.Vector()).hashCode()

Q51 Which statement regarding the following method definition is true?

```
boolean e() {
try {
assert false;
} catch (AssertionError ae) {
return true;
}
return false; // (1)
```

```
}
```

Select the one correct answer.

a. The code will fail to compile since catching an AssertionError is illegal.
b. The code will fail to compile since ther eturn statement at (1) is unreachable.
c. The method will return true under all circumstances.
d. The method will return false under all circumstances.
e. The method will return true if and only if assertions are enabled at runtime.

Q52 If str denotes a String object with the string "73", which of these expressions will convert the string to the int value 73? Select the two correct answers.

a. Integer.intValue(str)
b. ((int) str)
c. (new Integer(str)).intValue()
d. Integer.parseInt(str)
e. Integer.getInt(str)

Q53 Insert a line of code at the indicated location that will call the print() method in the Base class.

```
class Base {
public void print() {
System.out.println("base");
}
}
class Extension extends Base {
public void print() {
System.out.println("extension");
// Insert a line of code here.
}
}
public class Q294d {
public static void main(String[] args) {
Extension ext = new Extension();
ext.print();
}
}
```

Fill in a single line of code.

Q54 Given the following code, which statements are true?

```
public class Vertical {
private int alt;
public synchronized void up() {
++alt;
}
public void down() {
--alt;
}
public synchronized void jump() {
int a = alt;
up();
down();
assert(a == alt);
}
}
```

Select the two correct answers.

a. The code will fail to compile.
b. Separate threads can execute the up() method concurrently.
c. Separate threads can execute the down() method concurrently.
d. Separate threads can execute both the up() and down() method concurrently.
e. The assertion in the jump() method will not fail under any circumstances.

Q55 What will be written to the standard output when the following program is run?

```
public class Q03e4 {
public static void main(String[] args) {
```

```
String space = " ";
String composite = space + "hello" + space + space;
composite.concat("world");
String trimmed = composite.trim();
System.out.println(trimmed.length());
}
}
```

Select the one correct answer.

a. 5
b. 6
c. 7
d. 12
e. 13

Q56 Given the following code, which statements are true about the objects referenced through the fields i, j, and k, given that any thread may call the methods a(), b(), and c() at any time?

```
class Counter {
int v = 0;
synchronized void inc() { v++; }
synchronized void dec() { v--; }
}
public class Q7ed5 {
Counter i;
Counter j;
Counter k;
public synchronized void a() {
i.inc();
System.out.println("a");
i.dec();
}
public synchronized void b() {
i.inc(); j.inc(); k.inc();
System.out.println("b");
i.dec(); j.dec(); k.dec();
}
public void c() {
k.inc();
System.out.println("c");
k.dec();
}
}
```

Select the two correct answers.

a. i.v is guaranteed always to be 0 or 1.
b. j.v is guaranteed always to be 0 or 1.
c. k.v is guaranteed always to be 0 or 1
d. j.v will always be greater than or equal tok .v at any give time.
e. k.v will always be greater than or equal toj .v at any give time.

Q57 Which statements are true about casting and conversion? Select the three correct answers.

a. Conversion from int to long does not need a cast.
b. Conversion from byte to short does not need a cast.
c. Conversion from float to long does not need a cast.
d. Conversion from short to char does not need a cast.
e. Conversion from boolean to int using a cast is not possible.

Q58 Which method declarations, when inserted at the indicated position, will not cause the program to fail during compilation?

```
public class Qdd1f {
public long sum(long a, long b) { return a + b; }
// Insert new method declarations here.
}
```

Select the two correct answers.

a. public int sum(int a, int b) { return a + b; }
b. public int sum(long a, long b) { return 0; }
c. abstract int sum();
d. private long sum(long a, long b) { return a + b; }
e. public long sum(long a, int b) { return a + b; }

Q59 The 8859-1 character code for the uppercase letter A is the decimal value 65. Which code fragments declare and initialize a variable of type char with this value? Select the two correct answers.

a. char ch = 65;
b. char ch = '\65';
c. char ch = '\0041';
d. char ch = 'A';
e. char ch = "A";

Q60 What will be the result of executing the following program code with assertions enabled?

```
import java.util.*;
public class Q4d3f {
public static void main(String[] args) {
LinkedList lla = new LinkedList();
LinkedList llb = new LinkedList();
assert lla.size() == llb.size() : "empty";
lla.add("Hello");
assert lla.size() == 1 : "size";
llb.add("Hello");
assert llb.contains("Hello") : "contains";
assert lla.get(0).equals(llb.get(0)) : "element";
assert lla.equals(llb) : "collection";
}
}
```

Select the one correct answer.

a. Execution proceeds normally and produces no output.
b. An AssertionError with the message "size" is thrown.
c. An AssertionError with the message "empty" is thrown.
d. An AssertionError with the message "element" is thrown
e. An IndexOutOfBoundsException is thrown.
f. An AssertionError with the message "container" is thrown.

Q61 Which of these are keywords in Java? Select the two correct answers.

a. Double
b. native
c. main
d. unsafe
e. default

Mock Exam Answers

Q1 (a), (b), (c), and (e)
Only local variables need to be explicitly initialized before use. Fields are assigned a default value if not explicitly initialized.

Q2 *(a), (d), and (e)*

When the >> operator shifts bits to the right, it fills the new bits on the left with the bit value of the left-most bit of the original bit pattern. When the >>> operator shifts bits to the right, it always fills the new bits on the left with a bit value of 0. Thus, the >> and the >>> operators perform the same operation when the left-most bit of the original bit pattern has a bit value of 0. This occurs whenever the original value is non-negative.

The result of (-1 >> 1) is –1. A bit pattern consisting of all 1s (i.e., integer -1) will not change after being shifted by the >> operator.

The result of (-1 >>> 1) is 2147483647 which is $2^{31} - 1$. Shifting a bit pattern of all 1s one bit to the right using the >>> operator will yield a bit pattern of all 1s, except for the left-most bit which will be 0. This gives the non-negative value 2147483647, which is the maximum value of type int.

The >>> operator will shift 0s into the bit pattern from the left, thus giving the leftmost bit a value of 0. Since a value of 0 in the left-most bit signifies a non-negative value, the >>> operator is guaranteed to return a non-negative number if a shift has actually occurred.

Q3 (d)
A try block must be followed by at least one catch or finally block. No catch blocks can follow a finally block. Methods need not declare that they can throw Runtime Exceptions, as these are unchecked exceptions.

Q4 (e)
Giving parameters (2, 3) to the method substring() constructs a string consisting of the characters between positions 2 and 3 of the original string. The positions are indexed in the following manner: position 0 is immediately before the first character of the string, position 1 is between the first and the second character, position 2 is between the second and he third character, and so on.

Q5 (a)
A Thread object executes the run() method of a Runnable object on a separate thread when started. A Runnable object can be given when constructing a Thread object. If no Runnable object is supplied, the Thread object (which implements the Runnable interface) will execute its own run() method. A thread is initiated using the start() method of the Thread object.

Q6 (c)
The loop prints out the values 12, 9, 6, and 3 before terminating.

Q7 (e)
The fact that a field is static does not mean that it is not accessible from non-static methods and constructors. All fields are assigned a default value if no initializer is supplied. Local variables must be explicitly initialized before use.

Q8 (c)
The ^ operator will perform an XOR operation on the bit patterns 1001 and 0010, resulting in the bit pattern 1011, which will be written out in decimal form as 11.

Q9 (e)
All the "hello" literals denote the same String object. Any String object created using the new operator will be a distinct new object.

Q10 (d) and (e)

The main() method must be public and static, and take an array of String objects as parameter. It does not return a value and, therefore, should be declared void. The public and static modifiers must precede the keyword void. The (b) and (c) declarations will fail to compile due to error in the syntax of the method declaration. Declaration (a) will compile, but it does not meet the criteria of a main() method to start the execution of an application. Declaration (d) meets the criteria although the method is declared final, which is redundant since the method is static. It also uses a different parameter name, but it is none the less also a valid main() method. Declaration (e) is the canonical form of the main() method.

Q11 (a)
Calls to methods suspend(), sleep(), and wait() do not stop a thread. They only cause a thread to move out of its running state. A thread will terminate when the execution of the run() method has completed.

Q12 (d)
The Map interface provides operations that map keys to values.

Q13 (d)
The >>> operator clears the sign bit, so for negative values i >>> 1 is not equivalent to i/2.

Q14 (e)
An object of the class Extension is created. The first thing the constructor of Extension does is invoke the constructor of Base, using an implicit super() call. All calls to the method void add(int) are dynamically bound to the add() method in the Extension class, since the actual object is of type Extension. Therefore, this method is called by the constructor of Base, the constructor of Extension and the bogo() method with the parameters 1, 2, and 8, respectively. The instance field i changes value accordingly: 2, 6, and 22. The final value of 22 is printed.

Q15 (a) and (b)
The native modifier can be specified in the same position as accessibility modifiers in method declarations. Thus, the order of tokens in (a) and (b) is correct. The (c) declaration is rejected since the native modifier is not allowed after the declaration of the return type. Declaration (d) is rejected since it tries to declare an abstract method within a non-abstract class. The (e) declaration is rejected because native method declarations, just likea bstract method declarations, cannot have an implementation since this is defined elsewhere.

Q16 (d)
TreeMap and HashSet do not maintain an ordered sequence of objects. Vector and ArrayList require shifting of objects on insertion and deletion, while LinkedList does not. When objects are frequently inserted and deleted from the middle of the sequence, LinkedList gives the best performance.

Q17 (a) and (d)
Field b of the outer class is not shadowed by any local or inner class variables, therefore, (a) will work. Usintgh is.a will access the field a in the inner class. Using this.b will result in a compilation error since there is no field b in the inner class. Using Q4a39.this.a will successfully access the field of the outer class. The statement c = c will only reassign the current value of the local variable c to itself.

Q18 (d)
At (1), a new String object is constructed by concatenating the string "bye" in the String object denoted by b and the string "!". After line (2), d and b are aliases. After line (3), b and a are aliases, but d still denotes the String object with "bye" from line (0). After line (4), d and a are aliases. Reference d no longer denotes the String object created in line (0). This String object has no references to it and is, therefore, a candidate for garbage collection.

Q19 (d)
String objects are immutable. None of the methods of the String class modify a String object. Methods toUpperCase() and replace() in the String class will return a new String object that contains the modified string. However, StringBuffer objects are mutable.

Q20 (c)
Statement (a) will work just fine, and (b), (d), and (e) will cause compilation errors. Statements (b) and (e) will cause compilation errors since they attempt to assign an incompatible type to the reference. Statement

(d) will cause compilation errors since a cast from B to C is invalid. Being an instance of B excludes the possibility of being an instance of C. Statement (c) will compile, but will throw a runtime exception since the object that is cast toB is not an instance of B.

Q21 (a), (d), and (e)
String is a name of a class in the java.lang package, not a keyword. Java has a keyword null, but not NULL.

Q22 (e)
The desired accessibility is package accessibility, which is the default accessibility for members that have no accessibility modifier. The keyword package is not an accessibility modifier and cannot be used in this context.

Q23 (b) and (e)
For the expression tab[3][2] to access a valid element of a two-dimensional array, the array must have at least four rows and the fourth row must have at least three elements. Fragment (a) produces a 2 x 3 array. Fragment (c) tries to initialize a two-dimensional array as an one-dimensional array. Fragment (d) tries to specify array dimensions in the type of the array reference declaration.

Q24 (a)
The expression arr.length will evaluate to 4. The expression arr[1] will access the element { { "1", "2" }, { "1", null, "3" } }, and arr[1][2] will try to access the third sub-element of this element. This produces an ArrayIndexOutOfBoundsException, since the element has only two sub-elements.

Q25 (c) and (d)
String objects can have identical sequences of characters. The == operator, when used on String object references, will just compare the references and will only return true when both references denote the same object (i.e., are aliases). The equals() method will return true whenever the contents of the String objects are identical. An array of char and a String are two totally different types and cannot be compared using thee quals() method of the String class.

Q26 (b) and (c)
Statement (d) is false since an object of B can be created using the implicit default constructor of the class. B has an implicit default constructor since no constructor has explicitly been defined. Statement (e) is false since the second constructor of C will call the first constructor of C.

Q27 (b) and (d)
The retainAll(), removeAll(), and addAll() methods do not return a new collection object, but instead modify the collection object they were called upon. The collection object given as an argument is not affected. The containsAll() does not modify either of the collection objects.

Q28 (b), (c), and (e)
An instance of the class Baz is also an instance of the class Foo since the class Baz extends the class Foo. A Baz has a Bar since instances of the class Baz contain an instance of the class Bar by reference. A Foo has a Baz since instances of the class Foo contain an instance of the class Baz by reference. Since a Foo has a Baz which has a Bar, a
Foo has a Bar.

Q29 (b) and (e)
Unlike local variables, all fields are initialized with default initial values. All numeric fields are initialized to zero, boolean fields to false, char fields to '\u0000', and all reference fields to null.

Q30 (c) and (d)
Very little is guaranteed about the behavior of the garbage collection and finalization mechanisms. The (c) and (d) statements are two of the things that are guaranteed.

Q31 (a)
The main() method in (b) will always generate and catch an ArrayIndexOutOfBoundsException, since args.length is an illegal index in the args array. The main() method in (c) will always throw an ArrayIndexOutOfBoundsException since it is also uses args.length as an index, but this exception is never caught. The main() method in (d) will fail to print the argument if only one program argument is supplied.

The main() method in (e) will generate an uncaught ArrayIndexOutOfBoundsException if no program arguments are specified.

Q32 (a), (b), and (c)
Set and List both extend Collection. A map is not a collection and Map does not extend Collection. Set does not have any new methods other than those defined in Collection. List defines additional methods to the ones in Collection.

Q33 (c)
The type short defines 16-bit signed values in the range from -2^{15} to $2^{15}-1$, inclusive.

Q34 Filled in:
wait

Q35 (c)
The important thing to remember is that if an object is an instance of a class, then it is also an instance of all the superclasses of this class.

Q36 (c)
The expression (int) d evaluates to –2. The expression Math.ceil(d) evaluates to –2.0, giving the value –2 when converted to int. The expression Math.abs(d) evaluates to 2.9, giving the value 2 when converted to int.

Q37 (e)
Method g() modifies the field a. Method g() modifies the parameter b, not the field b, since the parameter declaration shadows the field. Variables are passed by value, so the change of value in parameter b is confined to the method g(). Method g() modifies the array whose reference value is passed as a parameter. Change to the first element is visible after return from the method g().

Q38 (a), (c), and (e)
The equals() method ignores the ordering of a and b when determining if two objects are equivalent. The hashCode() implementation must, therefore, also ignore the ordering of a and b when calculating the hash value, that is, the implementation must return the same value even after the values of a and b are swapped.

Q39 (c) and (d)
The variable j is only decremented if assertions are enabled. The assertion never fails, since the loop condition ensures that the loop body is only executed as long as i<j is true. With assertions enabled, each iteration decrements j by 1. The last number printed is 20 if assertions are disabled at runtime, and the last number printed is4 if assertions
are enabled at runtime.

Q40 (b) and (c)
These names have overloaded methods that allow optional timeout values as parameters.

Q41 (a), (b), and (d)
Both $ and _ are allowed as characters in identifiers. Character@ is not allowed in identifiers. All characters considered letters in the Unicode character set are allowed. The first character of an identifier cannot be a digit.

Q42 (a)
The program will fail to compile since the label l2 cannot precede the declaration int j = 0. For a label to be associated with a loop, it must immediately precede the loop construct.

Q43 (b)
Classes cannot extend interfaces, they must implement them. Interfaces can extend other interfaces, but cannot implement them. A class must be declared abstract if it does not provide an implementation for one of its methods. Methods declared in interfaces are implicitly public and abstract. Classes that implement these methods must explicitly declare their implementations public.

Q44 (a) and (e)
The notify() and notifyAll() methods are declared in the class Object. Since all other classes extend Object, these methods are also available in instances of all other classes, including Thread. The method notify() is not synchronized, but will throw an IllegalMonitorStateException if the current thread is not the owner of the object lock.

Q45 (a) and (c)
Accessing fields of the outer instance using only the variable name works within the inner instance as long as the variable is not shadowed. Fields need not be declared final in order to be accessible within the inner instance.

Q46 (a)
The code will fail to compile because the literal 4.3 has the type double. Assignment of a double value to a float variable without an explicit cast is not allowed. The code would compile and write 0 to standard output when run, if the literal 4.3 was replaced with 4.3F.

Q47 (b) and (e)
The && and || operators exhibit short-circuit behavior. The first operand of the ternary operator (? :) is always evaluated. Based on the result of this evaluation, either the second or the third operand is evaluated.

Q48 (e)
No labels are mandatory (including the default label) and can be placed in any order within thes witch body. The keyword continue may occur within the body of as witch statement as long as it pertains to a loop. Any constant nonlo- ng integral value can be used for case labels as long as the type is compatible with the expression in thes witch
expression.

Q49 (b) and (c)
The main() method must be declared static. It does not return a value and is, therefore, declared asv oid. It's accessibility modifier must be public. It has an array of String as parameter. Additionally it can be declared final, but that is not part of the requirement.

Q50 (a), (b), and (e)
Expressions (a), (b), and (e) all call the method hashCode() on valid objects. (c) is an illegal expression, as methods cannot be called on primitive values. The call in (d) to the equals() method requires an object as argument.

Q51 (e)
The method returns true if and only if assertions are enabled at runtime. It will only returnf alse if assertions are disabled.

Q52 (c) and (d)
(d) shows how the static method parseInt() of the wrapper class java.lang.Integer can be used to parse an integer from its string representation. It is also possible to first construct an Integer object based on the string representation and then extract the int value, using the intValue() method, as shown in (c).

Q53 Filled in:
super.print();
Overridden method implementations are accessed using the super keyword. Statements like print(), Base.print(), and Base.this.print() will not work.

Q54 (c) and (d)
Executing synchronized code does not guard against executing non-synchronized code concurrently.

Q55 (a)
Strings are immutable, therefore, the concat() method has no effect on the original String object. The string on which the trim() method is called consists of 8 characters, where the first and the two last characters are spaces" (hello "). The trim() method returns a new String object where the white space characters at each end have been removed. This leaves the 5 characters of the word "hello".

Q56 (a) and (b)
If a thread is executing method b() on an object, then it is guaranteed that no other thread executes methodsa () and b() concurrently. Therefore, the invocation counters i and j will never show more than one concurrent invocation. Two threads can concurrently be executing methods b() and c(). Therefore, the invocation counter k can easily show more than one concurrent invocation.

Q57 (a), (b), and (e)
Widening conversion allows conversions (a) and (b) to be done without a cast. Conversion from a floating point value to an integer value needs a cast. So does conversion from short values (which are signed) to char values (which are unsigned). Conversion from a boolean value to any other primitive type is not possible.

Q58 (a) and (e)
Declaration (b) fails since the method signature only differs in the return type. Declaration (c) fails since it tries to declare an abstract method in a non-abstract class. Declaration (d) fails since its signature is identical to the existing method.

Q59 (a) and (d)
The literal 65 is parsed as a value of type int. The value is within the range of the char type. A char literal can be specified by enclosing the character in single quotes ('A'), as a Unicode value in hexadecimal notation ('\u0041'), or as

an octal value ('\101'). The octal value cannot exceed '\377'. A char variable cannot be assigned a string, even if the string only contains one character.

Q60 (a)
Execution proceeds normally and produces no output. All assertions are true.

Q61 (b) and (e)
Note that keywords are case sensitive.