SCJP 1.6 (Sun Certified Java Programmer (SCJP) Mock Questions)

Final Test Questions

Questions: 71

Question - 1

What is the output for the below code ?

```
1. public  class A {
2.       int add(int i, int j){
3.              return i+j;
4.              }
5.}
6.public class B extends A{
7.      public static void main(String argv[]){
8.              short s = 9;
9.              System.out.println(add(s,6));
10.             }
11.}
```

Options are

A.Compile fail due to error on line no 2
B.Compile fail due to error on line no 9
C.Compile fail due to error on line no 8
D.15

Answer :
B is the correct answer.

Cannot make a static reference to the non-static method add(int, int) from the type A. The short s is autoboxed correctly, but the add() method cannot be invoked from a static method because add() method is not static.

Question - 2

What is the output for the below code ?

```
public class A {
        int k;
    boolean istrue;
    static int p;
        public void printValue() {
                System.out.print(k);
                System.out.print(istrue);
                System.out.print(p);

        }


}
public class Test{

    public static void main(String argv[]){

        A a = new A();

        a.printValue();
     }
}
```

Options are

A.0 false 0
B.0 true 0
C.0 0 0
D.Compile error - static variable must be initialized before use.

Answer :
A is the correct answer.

Global and static variable need not be initialized before use. Default value of global and static int variable is zero. Default value of boolean variable is false. Remember local variable must be initialized before use.

Question - 3

What is the output for the below code ?

```
public class Test{
        int _$;
        int $7;
        int do;
    public static void main(String argv[]){

      Test test = new Test();
      test.$7=7;
      test.do=9;

       System.out.println(test.$7);
       System.out.println(test.do);
       System.out.println(test._$);

     }
}
```

Options are

A.7 9 0
B.7 0 0
C.Compile error - $7 is not valid identifier.
D.Compile error - do is not valid identifier.

Answer :
D is the correct answer.

$7 is valid identifier. Identifiers must start with a letter, a currency character ($), or underscore (_). Identifiers cannot start with a number. You can't use a Java keyword as an identifier. do is a Java keyword.

Question - 4

What is the output for the below code ?

```
package com;
class Animal {

            public void printName(){
                        System.out.println("Animal");
            }

}
```

```
package exam;
import com.Animal;
public class Cat extends Animal {

            public void printName(){
                        System.out.println("Cat");
            }

}
```

```
package exam;
import com.Animal;

public class Test {

public static void main(String[] args){
            Animal a = new Cat();
            a.printName();

}

}
```

Options are

A.Animal
B.Cat
C.Animal Cat
D.Compile Error

Answer :
D is the correct answer.

Cat class won't compile because its superclass, Animal, has default access and is in a different package. Only public superclass can be accessible for different package.

Question - 5

```
What is the output for the below code ?
```

```
public  class A {
        int i = 10;
        public void printValue() {
```

```
                System.out.println("Value-A");
        };

}


public class B extends A{
        int i = 12;
        public void printValue() {
                System.out.print("Value-B");
        }
}


public class Test{

    public static void main(String argv[]){

        A a = new B();
        a.printValue();
        System.out.println(a.i);

     }
}
```

Options are

A.Value-B 11
B.Value-B 10
C.Value-A 10
D.Value-A 11

Answer :
B is the correct answer.

If you create object of subclass with reference of super class like (A a = new B();) then subclass method and super class variable will be executed.

Question - 6

```
What is the output for the below code ?

public enum Test {
        BREAKFAST(7, 30), LUNCH(12, 15), DINNER(19, 45);

        private int hh;

        private int mm;
```

```
        Test(int hh, int mm) {
                assert (hh >= 0 && hh <= 23) : "Illegal hour.";
                assert (mm >= 0 && mm <= 59) : "Illegal mins.";
                this.hh = hh;
                this.mm = mm;
        }

        public int getHour() {
                return hh;
        }

        public int getMins() {
                return mm;
        }

        public static void main(String args[]){
                Test t =  new BREAKFAST;
                System.out.println(t.getHour() +":"+t.getMins());
        }
}
```

Options are

A.7:30
B.Compile Error - an enum cannot be instantiated using the new operator.
C.12:30
D.19:45

Answer :
B is the correct answer.

As an enum cannot be instantiated using the new operator, the constructors cannot be called explicitly. You have to do like Test t = BREAKFAST;

Question - 7

```
What is the output for the below code ?

public class A {
        static{System.out.println("static");}
        { System.out.println("block");}
        public A(){
                System.out.println("A");
        }


public static void main(String[] args){
A a = new A();
```

```
}
```

Options are

A.A block static
B.static block A
C.static A
D.A

Answer :
B is the correct answer.

First execute static block, then statement block then constructor.

Question - 8

What is the output for the below code ?

```
1. public class Test {
2. public static void main(String[] args){
3.      int i = 010;
4.      int j = 07;
5.      System.out.println(i);
6.      System.out.println(j);
7. }
8. }
```

Options are

A.8 7
B.10 7
C.Compilation fails with an error at line 3
D.Compilation fails with an error at line 5

Answer :
A is the correct answer.

By placing a zero in front of the number is an integer in octal form. 010 is in octal form so its value is 8.

Question - 9

What is the output for the below code ?

```
1. public class Test {
2. public static void main(String[] args){
3.      byte b = 6;
4.      b+=8;
5.      System.out.println(b);
6.      b = b+7;
7.      System.out.println(b);
8.      }
9. }
```

Options are

A.14 21
B.14 13
C.Compilation fails with an error at line 6
D.Compilation fails with an error at line 4

Answer :
C is the correct answer.

int or smaller expressions always resulting in an int. So compiler complain about Type mismatch: cannot convert from int to byte for b = b+7; But b += 7; // No problem because +=, -=, *=, and /= will all put in an implicit cast. b += 7 is same as b = (byte)b+7 so compiler not complain.

Question - 10

What is the output for the below code ?

```
public class Test {
public static void main(String[] args){
        String value = "abc";
        changeValue(value);
        System.out.println(value);

}

public static void changeValue(String a){
        a = "xyz";
}

}
```

Options are

A.abc
B.xyz
C.Compilation fails
D.Compilation clean but no output

Answer :
A is the correct answer.

Java pass reference as value. passing the object reference, and not the actual object itself. Simply reassigning to the parameter used to pass the value into the method will do nothing, because the parameter is essentially a local variable.

Question - 11

What is the output for the below code ?

```
public class Test {

        public static void printValue(int i, int j, int k){
                System.out.println("int");
        }
        public static void printValue(byte...b){
                System.out.println("long");
        }

        public static void main(String... args) {
                byte b = 9;
                printValue(b,b,b);
        }
}
```

Options are

A.long
B.int
C.Compilation fails
D.Compilation clean but throws RuntimeException

Answer :
B is the correct answer.

Primitive widening uses the smallest method argument possible. (For Example if you pass short value to a method but method with short argument is not available then compiler choose method with int argument). But in this case compiler will prefer the older style before it chooses the newer style, to keep existing code more robust. var-args method is looser than widen.

Question - 12
Fill in the gap:

```
public class Test {


    public static void main(String[] args) {
        String[] words = new String[] {"aaa", "bbb", "ccc", "aaa"};
        Map<String, Integer> m = new TreeMap<String, Integer>();
        for (String word : words) {
             freq = m.get(word);
            m.put(word, freq == null ? 1 : freq + 1);
        }
        System.out.println(m);


    }
```

Use the following fragments zero or many times

String

Integer

Boolean

Float

Question - 13

You have a java file name Test.java inside src folder of javaproject directory.
You have also classes folder inside javaproject directory.

you have issued below command from command prompt.

```
cd javaproject
```

Which of the below command puts Test.class file inside classes folder ?

Options are

A.javac -d classes src/Test.java
B.javac Test.java
C.javac src/Test.java
D.javac classes src/Test.java

Answer :
A is the correct answer.

The -d option lets you tell the compiler in which directory to put the .class file it generates (d for destination)

Question - 14

```
You have two class files name Test.class and Test1.class inside
javaproject directory.

Test.java source code is :

public class Test{

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

Test1.java source code is :

public class Test1{

        public static void main (String[] args){
                System.out.println("Hello Test1");
        }
}


you have issued below commands from command prompt.

cd javaproject
java Test Test1

What is the output ?
```

Options are

A.Hello Test
B.Hello Test Hello Test1
C.Hello Test1
D.Run fails - class not found

Answer :
A is the correct answer.

You must specify exactly one class file to execute. If more than one then first one will be executed.

Question - 15

You have a java file name Test.java .

Test.java needs access to a class contained in app.jar in "exam" directory.

Which of the follwing command set classpath to compile clean?

Options are

A.javac -classpath exam/app.jar Test.java
B.javac -classpath app.jar Test.java
C.javac -classpath exam Test.java
D.None of the above

Answer :
A is the correct answer.

javac -classpath exam/app.jar Test.java is the correct command to set exam/app.jar in classpath.

Question - 16

What will be the result of compiling the following code:

```
public class SuperClass {
        public int doIt(String str, Integer... data)throws Exception{
                String signature = "(String, Integer[])";
                System.out.println(str + "  " + signature);
                return 1;
        }

}

public class SubClass extends SuperClass{

        public int doIt(String str, Integer... data)
        {
                String signature = "(String, Integer[])";
```

```
                System.out.println("Overridden: " + str + " " +
signature);
                return 0;
        }

        public static void main(String... args)
        {
                SuperClass sb = new SubClass();
                sb.doIt("hello", 3);
        }

}
```

Options are

A.Overridden: hello (String, Integer[])
B.hello (String, Integer[])
C.Complilation fails
D.None of the above

Answer :
C is the correct answer.

Unhandled exception type Exception.

Question - 17

```
What happens when the following code is compiled and run.
Select the one correct answer.

for(int i = 2; i < 4; i++)
  for(int j = 2; j < 4; j++)
    if(i < j)
       assert i!=j : i;
```

Options are

A.The class compiles and runs, but does not print anything.
B.The number 2 gets printed with AssertionError
C.compile error
D.The number 3 gets printed with AssertionError

Answer :
A is the correct answer.

When if condition returns true, the assert statement also returns true. Hence AssertionError does not get generated. .

Question - 18

```
What happens when the following code is compiled and run.
Select the one correct answer.

for(int i = 2; i < 4; i++)
  for(int j = 2; j < 4; j++)
     assert i!=j : i;
```

Options are

A.The class compiles and runs, but does not print anything.
B.The number 2 gets printed with AssertionError
C.compile error
D.The number 3 gets printed with AssertionError

Answer :
B is the correct answer.

When i and j are both 2, assert condition is false, and AssertionError gets generated. .

Question - 19

```
try{
                File f = new File("a.txt");
                        }catch(Exception e){

                        }catch(IOException io){

                        }
Is this code create new file name a.txt ?
```

Options are

A.True
B.False
C.Compilation Error
D.None

Answer :
C is the correct answer.

IOException is unreachable to compiler because all exception is going to catch by Exception block.

Question - 20

```
class A {
        A(String s) {
        }
        A() {
        }
}

1. class B extends A {
2.      B() { }
3.      B(String s) {
4.              super(s);
5.      }
6.      void test() {
7.              // insert code here
8.      }
9. }
```

Which of the below code can be insert at line 7 to make clean compilation ?

Options are

A.A a = new B();
B.A a = new B(5);
C.A a = new A(String s);
D.All of the above

Answer :
A is the correct answer.

A a = new B(); is correct because anonymous inner classes are no different from any other class when it comes to polymorphism.

Question - 21

What is the output for the below code ?

```
interface A {
        public void printValue();
```

```
}

1. public class Test{
2.      public static void main (String[] args){
3.              A a1 = new A() {
4.                      public void printValue(){
5.                              System.out.println("A");
6.                      }
7.                      };
8.                      a1.printValue();
9.                      }
10. }
```

Options are

A.Compilation fails due to an error on line 3
B.A
C.Compilation fails due to an error on line 8
D.null

Answer :
B is the correct answer.

The A a1 reference variable refers not to an instance of interface A, but to an instance of an anonymous (unnamed) class. So no compilation error.

Question - 22

```
class A {
        class A1 {
                void printValue(){
                        System.out.println("A.A1");
                }
        }
}

1. public class Test{
2.      public static void main (String[] args){
3.              A a = new A();
4.              // INSERT CODE
5.              a1.printValue();
6.      }
7. }
```

Which of the below code inserted at line 4, compile and produce the output "A.A1"?

Options are

A.A.A1 a1 = new A.A1();
B.A.A1 a1 = a.new A1();
C.A a1 = new A.A1();
D.All of the above

Answer :
B is the correct answer.

correct inner class instantiation syntax is A a = new A(); A.A1 a1 = a.new A1();

Question - 23

What is the output for the below code ?

```
public  class A {
        public void printValue(){
                System.out.println("Value-A");
        }
}

public class B extends A{
        public void printNameB(){
                System.out.println("Name-B");
        }
}

public class C extends A{

        public void printNameC(){
                System.out.println("Name-C");
        }

}

1. public class Test{
2.      public static void main (String[] args) {
3.              B b = new B();
4.              C c = new C();
5.              newPrint(b);
6.              newPrint(c);
7.      }
8.      public static void newPrint(A a){
9.              a.printValue();
10.     }
```

```
11. }
```

Options are

A.Value-A Name-B
B.Value-A Value-A
C.Value-A Name-C
D.Name-B Name-C

Answer :
B is the correct answer.

Class B extended Class A therefore all methods of Class A will be available to class B except private methods. Class C extended Class A therefore all methods of Class A will be available to class C except private methods.

Question - 24

```
What is the output for the below code ?

public  class A {
        public void printName(){
                System.out.println("Value-A");
        }
}

public class B extends A{
        public void printName(){
                System.out.println("Name-B");
        }
}

public class C extends A{

        public void printName(){
                System.out.println("Name-C");
        }

}

1. public class Test{
2.      public static void main (String[] args) {
3.              B b = new B();
4.              C c = new C();
5.              b = c;
6.              newPrint(b);
7.      }
```

```
8.      public static void newPrint(A a){
9.              a.printName();
10.     }
11. }
```

Options are

A.Name-B
B.Name-C
C.Compilation fails due to an error on lines 5
D.Compilation fails due to an error on lines 9

Answer :
C is the correct answer.

Reference variable can refer to any object of the same type as the declared reference OR can refer to any subtype of the declared type. Reference variable "b" is type of class B and reference variable "c" is a type of class C. So Compilation fails.

Question - 25

```
What is the output for the below code ?

public class C {

}

public class D extends C{

}

public class A {

        public C getOBJ(){
                System.out.println("class A - return C");
                return new C();

        }

}

public class B extends A{

        public D getOBJ(){
                System.out.println("class B - return D");
                return new D();

        }
```

```
}

public class Test {

public static void main(String... args) {
     A a = new B();
     a.getOBJ();

     }
}
```

Options are

A.class A - return C
B.class B - return D
C.Compilation fails
D.Compilation succeed but no output

Answer :
B is the correct answer.

From J2SE 5.0 onwards. return type in the overriding method can be same or subtype of the declared return type of the overridden (superclass) method.

Question - 26

What is the output for the below code ?

```
public  class A {
        private void printName(){
                System.out.println("Value-A");
        }
}

public class B extends A{
        public void printName(){
                System.out.println("Name-B");
        }
}


public class Test{
        public static void main (String[] args) {
                B b = new B();
                b.printName();
        }
```

```
}
```

Options are

A.Value-A
B.Name-B
C.Value-A Name-B
D.Compilation fails - private methods can't be override

Answer :
B is the correct answer.

You can not override private method , private method is not availabe in subclass . In this case printName() method a class A is not overriding by printName() method of class B. printName() method of class B different method. So you can call printName() method of class B.

Question - 27

```
What is the output for the below code ?

import java.io.FileNotFoundException;

public  class A {
        public void printName() throws FileNotFoundException {
                System.out.println("Value-A");
        }
}

public class B extends A{
        public void printName() throws NullPointerException{
                System.out.println("Name-B");
        }
}


public class Test{
        public static void main (String[] args) throws Exception{
                A a = new B();
                a.printName();
        }

}
```

Options are

A.Value-A
B.Compilation fails-Exception NullPointerException is not compatible with throws clause in A.printName()
C.Name-B
D.Compilation succeed but no output

Answer :
C is the correct answer.

The overriding method can throw any unchecked (runtime) exception, regardless of exception thrown by overridden method. NullPointerException is RuntimeException so compiler not complain.

Question - 28

What is the output for the below code ?

```
public  class A {
        public  A(){
                System.out.println("A");
        }
        public  A(int i){
                this();
                System.out.println(i);
        }
}


public class B extends A{
        public B (){
                System.out.println("B");
        }
        public B (int i){
                this();
                System.out.println(i+3);
        }
}


public class Test{

        public static void main (String[] args){
                new B(5);
        }
}
```

Options are

A.A B 8
B.A 5 B 8
C.A B 5
D.B 8 A 5

Answer :
A is the correct answer.

Constructor of class B call their superclass constructor of class A (public A()) , which execute first, and that constructors can be overloaded. Then come to constructor of class B (public B (int i)).

Question - 29

What is the output for the below code ?

```
1. public interface InfA {
2.              protected  String getName();
3. }

public class Test implements InfA{
        public  String getName(){
                return "test-name";
        }

        public static void main (String[] args){
                Test t = new Test();
                System.out.println(t.getName());

        }
}
```

Options are

A.test-name
B.Compilation fails due to an error on lines 2
C.Compilation fails due to an error on lines 1
D.Compilation succeed but Runtime Exception

Answer :
B is the correct answer.

Illegal modifier for the interface method InfA.getName(); only public and abstract are permitted

Question - 30

What is the output for the below code ?

```
public class D {
        int i;
        int j;
        public D(int i,int j){
                this.i=i;
                this.j=j;
        }

        public void printName()   {
                System.out.println("Name-D");
        }

}

1. public class Test{
2.      public static void main (String[] args){
3.              D d = new D();
4.              d.printName();
5.
6.      }
7. }
```

Options are

A.Name-D
B.Compilation fails due to an error on lines 3
C.Compilation fails due to an error on lines 4
D.Compilation succeed but no output

Answer :
B is the correct answer.

Since there is already a constructor in this class (public D(int i,int j)), the compiler won't supply a default constructor. If you want a no-argument constructor to overload the with-arguments version you already have, you have to define it by yourself. The constructor D() is undefined in class D. If you define explicit constructor then default constructor will

not be available. You have to define explicitly like public D(){ } then the above code will work. If no constructor into your class , a default constructor will be automatically generated by the compiler.

Question - 31

```
public  class A {
        public void test1(){
                System.out.println("test1");
        }
}

public class B extends A{
        public void test2(){
                System.out.println("test2");
        }
}

1. public class Test{
2.      public static void main (String[] args){
3.              A a = new A();
4.              A b = new B();
5.              B b1 = new B();
6.          // insert code here
7.      }
8. }
```

Which of the following , inserted at line 6, will compile and print test2?

Options are

A.((B)b).test2();
B.(B)b.test2();
C.b.test2();
D.a.test2();

Answer :
A is the correct answer.

((B)b).test2(); is proper cast. test2() method is in class B so need to cast b then only test2() is accessible. (B)b.test2(); is not proper cast without the second set of parentheses, the compiler thinks it is an incomplete statement.

Question - 32

What is the output for the below code ?

```
1. public class Test {
2.      public static void main(String... args) {
3.              int x =5;
4.              x *= 3 + 7;
5.              System.out.println(x);
6.      }
7. }
```

Options are

A.22
B.50
C.10
D.Compilation fails with an error at line 4

Answer :
B is the correct answer.

x *= 3 + 7; is same as x = x * (3 +7) = 5 * (10) = 50 because expression on the right is always placed inside parentheses.

Question - 33

What is the output for the below code ?

```
1. public class Test {
2.      enum Month { JAN, FEB, MAR };
3.      public static void main(String... args) {
4.              Month m1 = Month.JAN;
5.              Month m2 = Month.JAN;
6.              Month m3 = Month.FEB;
7.              System.out.println(m1 == m2);
8.              System.out.println(m1.equals(m2));
9.              System.out.println(m1 == m3);
10.             System.out.println(m1.equals(m3));
11.     }
12. }
```

Options are

A.true true true false
B.true true false false
C.false false true true
D.Compilation fails with an error at line 10

Answer :
B is the correct answer.

m1 and m2 refer to the same enum constant So m1 == m2 returns true BUT m1 and m3 refer to different enum constant So m1 == m3 returns false. m1.equals(m2) returns true because enum constant value is same (JAN and JAN). m1.equals(m3) return false because enum constants values are different (JAN and FEB).

Question - 34

What is the output for the below code ?

```
1. public class Test {
2.      public static void main(String... args) {
3.              int [] index = new int[5];
4.              System.out.println(index instanceof Object);
5.      }
6. }
```

Options are

A.true
B.false
C.Compilation fails with an error at line 3
D.Compilation fails with an error at line 4

Answer :
A is the correct answer.

An array is always an instance of Object

Question - 35

What is the output for the below code ?

```
public class Test {
```

```
        public static void main(String... args) {
                int a =5 , b=6, c =7;
                System.out.println("Value is "+ b +c);
                System.out.println(a + b +c);
                System.out.println("String "+(b+c));
        }
}
```

Options are

A.Value is 67 18 String 13
B.Value is 13 18 String 13
C.Value is 13 18 String
D.Compilation fails

Answer :
A is the correct answer.

If the left hand operand is not a String then + operator treat as plus BUT if left hand operand is a String then + perform String concatenation.

Question - 36

```
What is the output for the below code?
public class A {
        public A() {
         System.out.println("A");
    }
}

public class B extends A implements Serializable {
        public B() {
         System.out.println("B");
    }

}

public class Test {

        public static void main(String... args) throws Exception {
                B b = new B();

         ObjectOutputStream save = new ObjectOutputStream(new
FileOutputStream("datafile"));
         save.writeObject(b);
         save.flush();
```

```
        ObjectInputStream restore = new ObjectInputStream(new
FileInputStream("datafile"));
        B z = (B) restore.readObject();



        }

}
```

Options are

A.A B A
B.A B A B
C.B B
D.B

Answer :
A is the correct answer.

On the time of deserialization , the Serializable object not create new object. So constructor of class B does not called. A is not Serializable object so constructor is called.

Question - 37

```
What is the output for the below code?

public class A {
        public A() {
        System.out.println("A");
    }
}

public class Test {

        public static void main(String... args) throws Exception {
                A a = new A();

        ObjectOutputStream save = new ObjectOutputStream(new
FileOutputStream("datafile"));
        save.writeObject(a);
        save.flush();


        ObjectInputStream restore = new ObjectInputStream(new
FileInputStream("datafile"));
        A z = (A) restore.readObject();
```

```
        }

}
```

Options are

A.A A
B.A
C.java.io.NotSerializableException
D.None of the above

Answer :
C is the correct answer.

Class A does not implements Serializable interface. So throws NotSerializableException on trying to Serialize a non Serializable object.

Question - 38

What will be the result of compiling and run the following code:

```
public class Test {

        public static void main(String... args) throws Exception {
                Integer i = 34;
                int l = 34;
                if(i.equals(l)){
                        System.out.println(true);
                }else{
                        System.out.println(false);
                }

        }

}
```

Options are

A.true
B.false
C.Compile error
D.None of the above

Answer :
A is the correct answer.

equals() method for the integer wrappers will only return true if the two primitive types and the two values are equal.

Question - 39

```
What will be the result of compiling and run the following code:
public class Test {

        public static void main(String... args) throws Exception {
                File file = new File("test.txt");
                System.out.println(file.exists());
                file.createNewFile();
                System.out.println(file.exists());
        }

}
```

Options are

A.true true
B.false true
C.false true
D.None of the above

Answer :
B is the correct answer.

creating a new instance of the class File, you're not yet making an actual file, you're just creating a filename. So file.exists() return false. createNewFile() method created an actual file.so file.exists() return true.

Question - 40

```
What is the output for the below  code ?
public class A {}

public class B implements Serializable  {
        private static A a = new A();
        public static void main(String... args){
                B b = new B();
                try{
                        FileOutputStream fs = new
FileOutputStream("b.ser");
                        ObjectOutputStream os = new
ObjectOutputStream(fs);
```

```
                        os.writeObject(b);
                        os.close();

                }catch(Exception e){
                        e.printStackTrace();
                }

        }

}
```

Options are

A.Compilation Fail
B.java.io.NotSerializableException: Because class A is not Serializable.
C.No Exception at Runtime
D.None of the above

Answer :
C is the correct answer.

No java.io.NotSerializableException, Because class A variable is static. static variables are not Serializable.

Question - 41

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

```
1. public class Test extends Thread{
2.     public static void main(String argv[]){
3.       Test t = new Test();
4.       t.run();
5.       t.start();
6.     }
7.     public void run(){
8.       System.out.println("run-test");
9.     }
10. }
```

Options are

A.run-test run-test
B.run-test
C.Compilation fails due to an error on line 4

D.Compilation fails due to an error on line 7

Answer :
A is the correct answer.

t.run() Legal, but does not start a new thread , it is like a method call of class Test BUT t.start() create a thread and call run() method.

Question - 42

```
What is the output for the below code ?

class A implements Runnable{
        public void run(){
                System.out.println("run-a");
        }
}

1. public class Test {
2.      public static void main(String... args) {
3.              A a = new A();
4.              Thread t = new Thread(a);
5.              t.start();
6.              t.start();
7.      }
8. }
```

Options are

A.run-a
B.run-a run-a
C.Compilation fails with an error at line 6
D.Compilation succeed but Runtime Exception

Answer :
D is the correct answer.

Once a thread has been started, it can never be started again. 2nd time t.start() throws java.lang.IllegalThreadStateException.

Question - 43

What is the output for the below code ?

```
class A  implements Runnable{
        public void run(){
                try{
                for(int i=0;i<4;i++){
                        Thread.sleep(100);

        System.out.println(Thread.currentThread().getName());
                }
                }catch(InterruptedException e){

                }
        }
}


public class Test {
    public static void main(String argv[]) throws Exception{
      A a = new A();
      Thread t = new Thread(a,"A");
      Thread t1 = new Thread(a,"B");
      t.start();
      t.join();
      t1.start();
       }
}
```

Options are

A.A A A A B B B B
B.A B A B A B A B
C.Output order is not guaranteed
D.Compilation succeed but Runtime Exception

Answer :
A is the correct answer.

t.join(); means Threat t must finish before Thread t1 start.

Question - 44

What is the output for the below code ?

```
public class B {
        public synchronized void printName(){
                try{
```

```
                        System.out.println("printName");
                            Thread.sleep(5*1000);

                        }catch(InterruptedException e){

                        }
        }

        public  synchronized void printValue(){
                        System.out.println("printValue");

        }
}



public class Test extends Thread{
        B b = new B();
    public static void main(String argv[]) throws Exception{
    Test t = new Test();
    Thread t1 = new Thread(t,"t1");
    Thread t2 = new Thread(t,"t2");
    t1.start();
    t2.start();
    }

    public void run(){
        if(Thread.currentThread().getName().equals("t1")){
                b.printName();
        }else{
                b.printValue();
        }
    }

}
```

Options are

A.print : printName , then wait for 5 seconds then print : printValue
B.print : printName then print : printValue
C.print : printName then wait for 5 minutes then print : printValue
D.Compilation succeed but Runtime Exception

Answer :
A is the correct answer.

There is only one lock per object, if one thread has picked up the lock, no other thread can pick up the lock until the first thread releases the lock. printName() method acquire the lock for 5 seconds, So other threads can not access the object. If one synchronized method of an instance is executing then other synchronized method of the same instance should wait.

Question - 45

```
What is the output for the below code ?

public class B {
        public static synchronized void printName(){
                try{
                        System.out.println("printName");
                             Thread.sleep(5*1000);

                        }catch(InterruptedException e){

                        }
        }

        public synchronized void printValue(){
                        System.out.println("printValue");

        }
}



public class Test extends Thread{
        B b = new B();
    public static void main(String argv[]) throws Exception{
    Test t = new Test();
    Thread t1 = new Thread(t,"t1");
    Thread t2 = new Thread(t,"t2");
    t1.start();
    t2.start();
    }

    public void run(){
        if(Thread.currentThread().getName().equals("t1")){
                b.printName();
        }else{
                b.printValue();
        }
    }

}
```

Options are

A.print : printName , then wait for 5 seconds then print : printValue
B.print : printName then print : printValue
C.print : printName then wait for 5 minutes then print : printValue
D.Compilation succeed but Runtime Exception

Answer :
B is the correct answer.

There is only one lock per object, if one thread has picked up the lock, no other thread can pick up the lock until the first thread releases the lock. In this case printName() is static , So lock is in class B not instance b, both method (one static and other no-static) can run simultaneously. A static synchronized method and a non static synchronized method will not block each other.

Question - 46

What is the output for the below code ?

```
class A extends Thread{
        int count = 0;
                public void run(){
                        System.out.println("run");
                        synchronized (this) {
                                for(int i =0; i < 50 ; i++){
                                        count = count + i;
                                }
                                notify();
                }
        }
}



public class Test{

    public static void main(String argv[]) {
        A a = new A();
        a.start();
        synchronized (a) {
                System.out.println("waiting");
                try{
                        a.wait();
                }catch(InterruptedException e){

                }
                System.out.println(a.count);

                }

    }

}
```

Options are

A.waiting run 1225
B.waiting run 0
C.waiting run and count can be anything
D.Compilation fails

Answer :
A is the correct answer.

a.wait(); put thread on wait until not get notifed. A thread gets on this waiting list by executing the wait() method of the target object. It doesn't execute any further instructions until the notify() method of the target object is called. A thread to call wait() or notify(), the thread has to be the owner of the lock for that object.

Question - 47

Which of the following statements about this code are true?

```
class A  extends Thread{
        public void run(){
                for(int i =0; i < 2; i++){
                        System.out.println(i);
                }
        }
}


public class Test{
        public static void main(String argv[]){
                Test t = new Test();
                t.check(new A(){});
        }
        public void check(A a){
                a.start();
                }

}
```

Options are

A.0 0
B.Compilation error, class A has no start method
C.0 1
D.Compilation succeed but runtime exception

Answer :
C is the correct answer.

class A extends Thread means the anonymous instance that is passed to check() method has a start method which then calls the run method.

Question - 48

HashMap can be synchronized by _____ ?

Options are

A.Map m = Collections.synchronizeMap(hashMap);
B.Map m = hashMap.synchronizeMap();
C.Map m = Collection.synchronizeMap(hashMap);
D.None of the above

Answer :
A is the correct answer.

HashMap can be synchronized by Map m = Collections.synchronizeMap(hashMap);

Question - 49

What is the output for the below code?

```
import java.util.LinkedList;
import java.util.Queue;


public class Test {
        public static void main(String... args) {


                Queue q = new LinkedList();
                        q.add("newyork");
                        q.add("ca");
                        q.add("texas");
                        show(q);
                    }

                    public static void show(Queue q) {
                        q.add(new Integer(11));
                        while (!q.isEmpty ( ) )
                            System.out.print(q.poll() + "  ");
```

```
                    }

}
```

Options are

A.Compile error : Integer can't add
B.newyork ca texas 11
C.newyork ca texas
D.None of the above

Answer :
B is the correct answer.
" q was originally declared as Queue<String>, But in show() method it is passed as an untyped Queue. nothing in the compiler or JVM prevents us from adding an Integer after that.
 If the show method signature is public static void show(Queue<String> q) than you can't add Integer, Only String allowed. But public static void show(Queue q) is untyped Queue so you can add Integer.

Y poll() Retrieves and removes the head of this queue, or returns null if this queue is empty.

Question - 51

What is the output for the bellow code?

```
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;


public class Test {
        public static void main(String... args) {

                Set s = new TreeSet();
                  s.add("7");
                  s.add(9);
                  Iterator itr = s.iterator();
                    while (itr.hasNext())
                  System.out.print(itr.next() + " ");



        }

}
```

Options are

A.Compile error
B.Runtime Exception
C.7 9
D.None of the above

Answer :
B is the correct answer.
Without generics, the compiler does not know what type is appropriate for this TreeSet, so it allows everything to compile. But at runtime he TreeSet will try to sort the elements as they are added, and when it tries to compare an Integer with a String it will throw a ClassCastException.
? Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer.

Question - 52

```
What is the output for the below code?

import java.util.Iterator;
import java.util.TreeSet;


public class Test {
        public static void main(String... args) {

                TreeSet s1 = new TreeSet();
                s1.add("one");
                s1.add("two");
                s1.add("three");
                s1.add("one");

                    Iterator it = s1.iterator();
                   while (it.hasNext() ) {
                        System.out.print( it.next() + " " );
                    }



        }

}
```

Options are

A.one three two
B.Runtime Exception
C.one three two one
D.one two three

Answer :
A is the correct answer.
h TreeSet assures no duplicate entries.it will return elements in natural order, which, for Strings means alphabetical.

Question - 53

```
If we do

ArrayList lst = new ArrayList();

What is the initial capacity of the ArrayList lst ?
```

Options are

A.10
B.8
C.15
D.12

Answer :
A is the correct answer.

/** * Constructs an empty list with an initial capacity of ten. */ public ArrayList()
{ this(10); }

Question - 54

```
What is the output for the below code ?

package bean;

public class Abc {
 public static int index_val = 10;
}

package com;
import static bean.Abc.index_val;

public class Test1 {

        public static void main(String... args) {
             System.out.println(index_val);
        }
}
```

Options are

A.10
B.compile error, index_val not defined
C.Compile error at import static bean.Abc.index_val;
D.None of the above

Answer :
A is the correct answer.

The static import construct allows unqualified access to static members without inheriting from the type containing the static members. J2SE 5.0 onwards it allows static import like import static bean.Abc.index_val; and can be use directly System.out.println(index_val);

Question - 55

Which of the following statement is true about jar command?

Options are

A.The jar command creates the META-INF directory implicitly.
B.The jar command creates the MANIFEST.MF file implicitly.
C.The jar command would not place any of your files in META-INF directory.
D.All of the above are true

Answer :
A is the correct answer.

All statements are true.

Question - 56

You have a class file name Test.class inside javaproject directory.

Test.java source code is :

```
import java.util.Properties;
class Test {
        public static void main (String[] args){
                Properties p = System.getProperties();
                System.out.println(p.getProperty("key1"));
        }
}
```

```
you have issued below commands from command prompt.

cd javaproject
java -D key1=value1 Test

What is the output ?
```

Options are

A.value1
B.null
C.Run successfully but no output
D.Run fails - java.lang.NoClassDefFoundError: key1=value1

Answer :
D is the correct answer.

-D option , name=value pair must follow immediately, no spaces allowed. In this case there is space between -D and key1=value1 So java.lang.NoClassDefFoundError: key1=value1.

Question - 57

```
What is the output for the below code ?

public class A {
   public void printValue(){
           System.out.println("A");
   }
}

public class B extends A {
        public void printValue(){
                   System.out.println("B");
           }
}

1. public class Test {
2.      public static void main(String... args) {
3.              A b = new B();
4.              newValue(b);
5.      }
6.      public static void newValue(A a){
7.              if(a instanceof B){
8.                      ((B)a).printValue();
9.              }
10.     }
```

```
11. }
```

Options are

A.A
B.B
C.Compilation fails with an error at line 4
D.Compilation fails with an error at line 8

Answer :
B is the correct answer.

instanceof operator is used for object reference variables to check whether an object is of a particular type. In newValue(b); b is instance of B So works properly.

Question - 58

What is the output for the below code ?

```
1. public class Test {
2.      static int i =5;
3.      public static void main(String... args) {
4.              System.out.println(i++);
5.              System.out.println(i);
6.              System.out.println(++i);
7.              System.out.println(++i+i++);
8.
9.      }
10. }
```

Options are

A.5 6 7 16
B.6 6 6 16
C.6 6 7 16
D.5 6 6 16

Answer :
A is the correct answer.

i++ : print value then increment (postfix - increment happens after the value of the variable is used) ++i : increment the print (prefix - increment happens before the value of the variable is used)

Question - 59

```
What is the output for the below code ?

1. public class Test {
2.      public static void main(String... args) {
3.              Integer i = 34;
4.              String str = (i<21)?"jan":(i<56)?"feb":"march";
5.              System.out.println(str);
6.      }
7. }
```

Options are

A.feb
B.jan
C.march
D.Compilation fails with an error at line 4

Answer :
A is the correct answer.

This is nested conditional with unbox. (i<21) is false goto (i<56), (i<56) is true so result is "feb".

Question - 60

```
What is the output ?

public class Test {

        public static void main(String... args) {

                Pattern p = Pattern.compile("a+b?c*");
                Matcher m = p.matcher("ab");
                boolean b = m.matches();
                System.out.println(b);



        }
}
```

Options are

A.true

B.false
C.Compile error
D.None of the above

Answer :
A is the correct answer.

X? X, once or not at all X* X, zero or more times X+ X, one or more times

Question - 61

What is the output for the below code ?

```
1. public class Test {
2. public static void main(String[] args){
3.      byte i = 128;
4.      System.out.println(i);
5.      }
6. }
```

Options are

A.128
B.0
C.Compilation fails with an error at line 3
D.Compilation fails with an error at line 4

Answer :
C is the correct answer.

byte can only hold up to 127. So compiler complain about possible loss of precision.

Question - 62

What is the output for the below code ?

```
1. public class Test {
2.      int i=8;
3.      int j=9;
4. public static void main(String[] args){
5.      add();
6. }
7.   public static void add(){
8.         int k = i+j;
```

```
9.          System.out.println(k);
10.    }
11. }
```

Options are

A.17
B.0
C.Compilation fails with an error at line 5
D.Compilation fails with an error at line 8

Answer :
D is the correct answer.

i and j are instance variable and attempting to access an instance variable from a static method. So Compilation fails .

Question - 63

Which collection class grows or shrinks its size and provides indexed access to its elements, but methods are not synchronized?

Options are

A.java.util.ArrayList
B.java.util.List
C.java.util.HashSet
D.java.util.Vector

Answer :
A is the correct answer.
ArrayList provides an index to its elements and methods are not synchronized.

Question - 64

What is the output of bellow code ?

```
public class Bean{
        private String str;

        Bean(String str ){
                this.str = str;
        }

        public String getStr() {
                return str;
```

```
        }

        public boolean equals(Object o){
                if (!(o instanceof Bean)) {
                        return false;
                }

                return ((Bean) o).getStr().equals(str);
        }

        public int hashCode() {
                return 12345;
        }

        public String toString() {
                return str;
        }
}



import java.util.HashSet;
public class Test {
        public static void main(String ... sss) {
                HashSet myMap = new HashSet();
                String s1 = new String("das");
                String s2 = new String("das");
                Bean s3 = new Bean("abcdef");
                Bean s4 = new Bean("abcdef");

                myMap.add(s1);
                myMap.add(s2);
                myMap.add(s3);
                myMap.add(s4);

                System.out.println(myMap);
        }
}
```

Options are

A.das abcdef
B.das abcdef das abcdef
C.das das abcdef abcdef
D.das

Answer :
A is the correct answer.

implemented 'equals' and 'hashCode' methods to get unique result in Set.

Question - 65

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

```
class A  implements Runnable{
        public void run(){
                System.out.println("run-A");
        }
}

1. public class Test {
2.    public static void main(String argv[]){
3.      A a = new A();
4.      Thread t = new Thread(a);
5.     System.out.println(t.isAlive());
6.     t.start();
7.     System.out.println(t.isAlive());
8.    }
9. }
```

Options are

A.false run-A true
B.false run-A false
C.true run-A true
D.Compilation fails due to an error on line 7

Answer :
A is the correct answer.

Once the start() method is called, the thread is considered to be alive.

Question - 66

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

```
1. public class Test extends Thread{
2.    public static void main(String argv[]){
3.      Test t = new Test();
4.      t.run();
5.      t.start();
6.    }
7.    public void run(){
```

```
8.      System.out.println("run-test");
9.    }
10. }
```

Options are

A.run-test run-test
B.run-test
C.Compilation fails due to an error on line 4
D.Compilation fails due to an error on line 7

Answer :
A is the correct answer.

t.run() Legal, but does not start a new thread , it is like a method call of class Test BUT t.start() create a thread and call run() method.

Question - 67

Which of the following are methods of the Thread class?

```
1) yield()
2) sleep(long msec)
3) go()
4) stop()
```

Options are

A.1 , 2 and 4
B.1 and 3
C.3 only
D.None of the above

Answer :
A is the correct answer.

Check out the Java2 Docs for an explanation

Question - 68

What notifyAll() method do?

Options are

A.Wakes up all threads that are waiting on this object's monitor
B.Wakes up one threads that are waiting on this object's monitor
C.Wakes up all threads that are not waiting on this object's monitor
D.None of the above

Answer :
A is the correct answer.

notifyAll() : Wakes up all threads that are waiting on this object's monitor.A thread waits on an object's monitor by calling one of the wait methods.

Question - 69

What is the output for the below code?

```
import java.util.NavigableMap;
import java.util.concurrent.ConcurrentSkipListMap;


public class Test {
        public static void main(String... args) {


                NavigableMap navMap = new
         ConcurrentSkipListMap();

                navMap.put(4, "April");
                navMap.put(5, "May");
                navMap.put(6, "June");
                navMap.put(1, "January");
                navMap.put(2, "February");
                navMap.put(3, "March");

         navMap.pollFirstEntry();
         navMap.pollLastEntry();
         navMap.pollFirstEntry();
         System.out.println(navMap.size());



         }
}
```

Options are

A.Compile error : No method name like pollFirstEntry() or pollLastEntry()
B.3
C.6
D.None of the above

Answer :
B is the correct answer.
Y pollFirstEntry() Removes and returns a key-value mapping associated with the least key in this map, or null if the map is empty.
Y pollLastEntry() Removes and returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.

Question - 70

What is the output for the bellow code?

```
import java.io.Console;

public class Test {
        public static void main(String... args) {


                Console con = System.console();
                  boolean auth = false;

                  if (con != null)
                  {
                    int count = 0;
                    do
                    {
                      String uname = con.readLine(null);
                      char[] pwd = con.readPassword("Enter %s's
password: ", uname);

                      con.writer().write("\n\n");
                    } while (!auth && ++count < 3);
                  }



        }
}
```

Options are

A.NullPointerException
B.It works properly
C.Compile Error : No readPassword() method in Console class.
D.None of the above

Answer :
A is the correct answer.
$ passing a null argument to any method in Console class will cause a NullPointerException to be thrown.

Question - 71

```
What is the output for the below code?

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.NavigableSet;
import java.util.TreeSet;


public class Test {
        public static void main(String... args) {


                List  lst = new ArrayList ();
                lst.add(34);
                lst.add(6);
                lst.add(2);
                lst.add(8);
             lst.add(7);
             lst.add(10);

                NavigableSet   nvset = new TreeSet(lst);
                 System.out.println(nvset.headSet(10,true));


        }
}
```

Options are

A.Compile error : No method name like headSet()
B.2, 6, 7, 8, 10
C.2, 6, 7, 8
D.None of the above

Answer :
B is the correct answer.
headSet(10) Returns the elements elements are strictly less than 10.
q headSet(10,false) Returns the elements elements are strictly less than 10.
- headSet(10,true) Returns the elements elements are strictly less than or equal to 10.

Question - 72

```
What is the output?

import java.util.ArrayList;
import java.util.List;
import java.util.NavigableSet;
```

```
import java.util.TreeSet;


public class Test {
        public static void main(String... args) {


                List lst = new ArrayList();
                lst.add(34);
                lst.add(6);
                lst.add(2);
                lst.add(8);
                 lst.add(7);
                 lst.add(10);

               NavigableSet  nvset = new TreeSet(lst);
               System.out.println(nvset.lower(6)+" "+nvset.higher(6)+ "
"+ nvset.lower(2));

        }
}
```

Options are

A.1 2 7 10 34 null
B.2 7 null
C.2 7 34
D.1 2 7 10 34

Answer :
B is the correct answer.
lower() Returns the greatest element in this set strictly less than the given element, or null if there is no such element.
higher() Returns the least element in this set strictly greater than the given element, or null if there is no such element.