1.

­Consider the following code snippet:

XXXX m ;

switch( m )

{

case 32 : System.out.println("32"); break;

case 64 : System.out.println("64"); break;

case 128 : System.out.println("128"); break;

}

What type can 'm' be of so that the above code compiles and runs as expected ?

Select 3 correct options

a int m;

m can hold all the case values.

b long m;

long, float, double and boolean can never be used as a switch variable.

c char m;

m can hold all the case values.

d byte m;

m will not be able to hold 128. a byte's range is -128 to 127.

e short m;

m can hold all the case values.

Ans: a,c,e

2.

Consider the following code:

interface I

{

int i = 1, ii = Test.out("ii", 2);

}

interface J extends I

{

int j = Test.out("j", 3), jj = Test.out("jj", 4);

}

interface K extends J

{

int k = Test.out("k", 5);

}

class Test

{

public static void main(String[] args) { System.out.println(K.j); }

public static int out(String s, int i)

{

System.out.println(s + "=" + i);

return i;

}

}

What will be the output when class Test is run ?

Select 1 correct option.

a It will print ii=2, j = 3, jj=4, k=5 and then 3.

b It will print j = 3, jj=4 and then 3.

c It will print j=3 and then 3.

d It will not even compile.

e None of the above.

Ans: b

3.

Which of these is the correct format to use to create the char literal of value a?

Select 2 correct options

a 'a'

b "a"

It's a String object.

c new Character(a)

It's a Character object.

d \u00061

It represents 'a' but not as char literal but as a variable.

e '\u0061'

ans: a,e

4.

What will be the output of the following class:

public class TestClass

{

public void testRefs(String str, String sb)

{

str = str + sb.toString();

str = null;

sb = null;

}

public static void main(String[] args)

{

String s = "aaa";

String sb = "bbb";

new TestClass().testRefs(s, sb);

System.out.println("s="+s+" sb="+sb);

}

}

Select 1 correct option.

a s=aaa sb=bbb

b s=null sb=null

c s=aaa sb=null

d s=null sb=bbbaaa

e s=aaa sb=bbbaaabbb

ans: a

5.

What will the following code print?

int i = 0xFFFFFFF2;

int j = ~i;

System.out.println(j);

Select 1 correct option.

a -14

b 13

c -13

d 11

e -11

ans: b

6.

Any class may be unloaded when none of it's instances and class objects that represent this class are reachable.

Select 1 correct option.

a True

b False

ans: b

General Comments

A class or interface may be unloaded if and only if its class loader is unreachable (the definition of unreachable is given in JLS 12.6.1). Classes loaded by the bootstrap loader are not unloaded.

7.

Consider the following class and interface definitions:

public class Sample implements IInt

{

public static void main(String[] args)

{

Sample s = new Sample(); //1

int j = s.thevalue; //2

int k = IInt.thevalue; //3

int l = thevalue; //4

}

}

public interface IInt

{

int thevalue = 0;

}

What will happen when the above code is compiled and run?

Select 1 correct option.

a It will give an error at compile time at line //1.

b It will give an error at compile time at line //2.

c It will give an error at compile time at line //3

d It will give an error at compile time at line //4.

e It will compile and run without any problem.

ans: e

General Comments

As a rule, fields defined in an interface are public, static, and final.

Here, the interface IInt defines 'thevalue' and thus any class that implements this interface inherits this field. Therefore, it can be accessed using s.thevalue or just 'thevalue' inside the class. Also, since it is static, it can also be accessed using IInt.thevalue or Sample.thevalue.

8.

Consider the following lines of code:

Integer i = new Integer(42);

Long ln = new Long(42);

Double d = new Double(42.0);

Which of the following options are valid?

Select 2 correct options

a i == ln;

This will fail at compile time

b ln == d;

This will fail at compile time

c i.equals(d);

d d.equals(ln);

e ln.equals(42);

You cannot pass primitives to equals() method. Only objects can be passed

Ans: c,d

General Comments

The concept to understand here is:

If the compiler can figure out that something can NEVER happen, then it flags an error. In this question, the compiler knows that ln, i or d can never point to the same object in any case as they are references to different class of objects which have no relation ( superclass/subclass ) between themselves.

9.

What will the following program print?

public class TestClass

{

static String str = "Hello World";

public static void changeIt(String s)

{

s = "Good bye world";

}

public static void main(String[] args)

{

changeIt(str);

System.out.println(str);

}

}

Select 1 correct option.

a "Hello World"

b "Good bye world"

c It will not compile.

d It will throw an exception at runtime.

e None of the above.

Ans: a

10.

Consider the following code snippet ...

boolean[] b1 = new boolean[2];

boolean[] b2 = {true , true};

System.out.println( "" + (b1[0] == b2[0]) + ", "+ (b1[1] == b2[1]) );

What will it print ?

Select 1 correct option.

a It will not compile.

b It will throw ArrayIndexOutOfBoundsError at Runtime.

c It will print false, true.

d It will print true, false.

e It will print false, false.

Ans: e

11.

What will the following program print?

public class TestClass

{

public static void main(String[] args)

{

unsigned byte b = 0;

b--;

System.out.println(b); }

}

Select 1 correct option.

A 0

B -1

C 255

D -128

E It will not compile.

ANS : e

12.

What will be the output of the following program ?

class Test

{

public static void main(String[] args)

{

String s = "going";

print(s, s = "gone");

}

static void print(String a, String b)

{

System.out.println(a +", "+ b );

}

}

Select 1 correct option.

a It will print going, gone

b It will print gone, gone.

c It will print gone, going.

d It will print going, going.

e It will not compile.

Ans: a

The assignment of the string "gone" to s occurs after the first argument to print has been evaluated.

If evaluation of an argument expression completes abruptly, no part of any argument expression to its right appears to have been evaluated.

13.

What will the following code snippet print:

Float f = null;

try

{

f = Float.valueOf("12.3");

String s = f.toString();

int i = Integer.parseInt(s);

System.out.println("i = "+i);

}

catch(Exception e)

{

System.out.println("trouble : "+f);

}

Select 1 correct option.

a 12

b 13

c trouble : null

d trouble : 12.3

e trouble : 0.0

ans: d

14.

Which of the following are true about the "default" constructor?

Select 2 correct options

a It is provided by the compiler only if the class does not define any constructor.

b It initializes the instance members of the class.

c It calls the default 'no-args' constructor of the super class.

d It initializes instance as well as class fields of the class.

e It is provided by the compiler if the class does not define a 'no- args' constructor.

It is not provided even if the class declares any other constructor.

Ans: a,c

15.

Which of the following lines can be inserted at line 1 to make the program run?

//line 1

public class TestClass

{

public static void main(String[] args)

{

PrintWriter pw = new PrintWriter(System.out);

OutputStreamWriter osw = new OutputStreamWriter( System.out);

pw.print("hello");

}

}

Select 1 correct option.

a import java.lang.\*;

Although you can imort it, it is automatically imported by the compiler.

b import java.io.\*;

This will make all the classes of io package available.

c import java.io.OutputStreamWriter;

This will only make OutputStreamwriter available. PrintWriter will still be unavailable.

d include java.io.\*;

This is not C/C++ !

e include java.lang.System

ans: b