Java Examen

Puntos totales 18/53

Correo *

armando.molina@accenture.com

✗ What is the result? * 0/1

```
public class Bees {
        public static void main(String[] args) {
                try {
                        new Bees().go();
                } catch (Exception e) {
                        System.out.println("thrown to main");
                }
        }
        synchronized void go() throws InterruptedException {
                Thread t1 = new Thread();
                t1.start();
                System.out.print("1 ");
                t1.wait(5000);
                System.out.print("2 ");
        }
}
```

- (●) The program prints 1 then 2 after 5 seconds. ✗
- ( ) The program prints: 1 thrown to main.
- ( ) The program prints: 1 2 thrown to main.
- ( ) The program prints:1 then t1 waits for its notification.

Se tardó mucho tiempo en procesar por lo cual te manda con la excepción. ¿En este caso me pregunto entonces como le haces para que el código si es muy pesado se espere?

Respuesta correcta

- (●) The program prints: 1 thrown to main.

✓ Which two possible outputs? * 1/1

```
public class Main {
        public static void main(String[] args) throws Exception {
                doSomething();
        }
        private static void doSomething() throws Exception {
                System.out.println("Before if clause");
                if (Math.random() > 0.5) { throw new Exception();}
                System.out.println("After if clause");
        }
}
```

- [x] Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15). ✓
- [ ] Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15) After if clause.
- [ ] Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15).
- [x] Before if clause After if clause. ✓

✓ What is the result? * 1/1

```
public class Test {
        public static void main(String[] args) {
                int[][] array = { {0}, {0,1}, {0,2,4}, {0,3,6,9}, {0,4,8,12,16} };
                System.out.println(array[4][1]);
                System.out.println(array[1][4]);
        }
}
```

- 4 Null.
- Null 4.
- An IllegalArgumentException is thrown at run time.
- (●) 4 An ArrayIndexOutOfBoundsException is thrown at run time. ✓

✓ Whats is the result? * 1/1

```
public static void main(String[] args) {
        System.out.println("Result:" + 3 + 5);
        System.out.println("Result:" + (3 + 5));
}
```

- Result: 8 Result: 8
- (●) Result: 35 Result: 8 ✓
- Result: 8 Result: 35
- Result: 35 Result: 35

✗ What is the result? * 0/1

```
class MySort implements Comparator<Integer> {
        public int compare(Integer x, Integer y) {
                return y.compareTo(x);
        }
}
```

And the code fragment:

```
Integer[] primes = {2, 7, 5, 3};
MySort ms = new MySort();
Arrays.sort(primes, ms);
for (Integer p2 : primes) { System.out.print(p2 + " "); }
```

- (●) 2 3 5 7 ✗
- ( ) 2 7 5 3
- ( ) 7 5 3 2
- ( ) Compilation fails.

Respuesta correcta

- (●) 7 5 3 2

✓ What value of x, y, z will produce the following result? *1/1

1234,1234,1234 -----, 1234, ------

```
public static void main(String[] args) {
        // insert code here
        int j = 0, k = 0;
        for (int i = 0; i < x; i ++) {
                do {
                        k = 0;
                        while (k < z) {
                                k++;
                                System.out.print(k + " ");
                        }
                        System.out.println(" ");
                        j++;
                } while (j < y);
                System.out.println("---");
        }
}
```

- ○ int x = 4, y = 3, z = 2;
- ○ int x = 3, y = 2, z = 3;
- ○ int x = 2, y = 3, z = 3;
- ○ int x = 4, y = 2, z= 3;
- ◉ int x = 2, y = 3, z = 4; ✓

✓ What is the result? * 1/1

```
public class Test {
        public static void main(String[] args) {
                int b = 4;
                b--;
                System.out.print(--b);
                System.out.println(b);
        }
}
```

- (●) 22 ✓
- ( ) 12
- ( ) 32
- ( ) 33

✓ What is the result? * 1/1

```
int a = 10;
int b = 37;
int z = 0;
int w = 0;
if (a == b) { z = 3; } else if (a > b) { z = 6; }
w = 10 * z;
```

- (●) 0 ✓
- ( ) 30
- ( ) 60

✓ What is the result? * 1/1

```
public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        System.arraycopy(array, 2, array, 1, 2);
        System.out.print(array[1]);
        System.out.print(array[4]);
}
```

- ○ 14
- ○ 15
- ○ 24
- ○ 25
- ○ 34
- ◉ 35 ✓

✗ What is the result? * 0/1

```
class Feline {
        public String type = "f ";
        public Feline() {
                System.out.print("feline ");
        }
}
public class Cougar extends Feline {
        public Cougar() {
                System.out.print("cougar ");
        }
        void go() {
                type = "c ";
                System.out.print(this.type + super.type);
        }
        public static void main(String[] args) {
                new Cougar().go();
        }
}
```

- ○ Cougar c f.
- ○ Feline cougar c c.
- ◉ Feline cougar c f. ✗
- ○ Compilation fails.

Respuesta correcta

- ◉ Feline cougar c c.

✓ Which statement is true? * 1/1

```
class ClassA {
        public int numberOfInstances;
        protected ClassA(int numberOfInstances) {
                this.numberOfInstances = numberOfInstances;
        }
}
public class ExtendedA extends ClassA {
        private ExtendedA(int numberOfInstances) {
                super(numberOfInstances);
        }
        public static void main(String[] args) {
                ExtendedA ext = new ExtendedA(420);
                System.out.print(ext.numberOfInstances);
        }
}
```

- (●) 420 is the output. ✓
- ( ) An exception is thrown at runtime.
- ( ) All constructors must be declared public.
- ( ) Constructors CANNOT use the private modifier.
- ( ) Constructors CANNOT use the protected modifier.

✓ What is the result? * 1/1

```
class X {
        static void m(int i) {
                i += 7;
        }
        public static void main(String[] args) {
                int i = 12;
                m(i);
                System.out.println(i);
        }
}
```

- ○ 7
- ◉ 12 ✓
- ○ 19
- ○ Compilation fails.
- ○ An exception is thrown at run time

✓ What changes will make this code compile? * 1/1

```
class X {
        X() { }
        private void one() { }
}
public class Y extends X {
        Y() { }
        private void two() {
        one();
        }
        public static void main(String[] args) {
        new Y().two();
        }
}
```

- ( ) Adding the public modifier to the declaration of class X.
- ( ) Adding the protected modifier to the X() constructor.
- (•) Changing the private modifier on the declarationof the one() method to protected. ✓
- ( ) Removing the Y() constructor.

✓ The pom.xml file is the configuration file For: * 1/1

- ( ) Ant
- ( ) Gradle
- (•) Maven ✓

✓ Whats is true about the class Wow? * 1/1

```
public abstract class Wow {
        private int wow;
        public Wow(int wow) { this.wow = wow; }
        public void wow() { }
        private void wowza() { }
}
```

- (●) It compiles without error. ✓
- ( ) It does not compile because an abstract class cannot have private methods
- ( ) It does not compile because an abstract class cannot have instance variables.
- ( ) It does not compile because an abstract class must have at least one abstract method.
- ( ) It does not compile because an abstract class must have a constructor with no arguments.

✓ How many times is 2 printed? * 1/1

```
public static void main(String[] args) {
        String[] table = {"aa", "bb", "cc"};
        int ii = 0;
        for (String ss : table) {
                while (ii < table.length) {
                        System.out.println(ii); ii++;
                        break;
                }
        }
}
```

- ○ Zero.
- ◉ Once. ✓
- ○ Twice.
- ○ Thrice.
- ○ It is not printed because compilation fails.

✓ What is printed out when the program is excuted? * 1/1

```
public class MainMethod {
        void main() {
                System.out.println("one");
        }
        static void main(String args) {
                System.out.println("two");
        }
        public static final void main(String[] args) {
                System.out.println("three");
        }
        void mina(Object[] args) {
                System.out.println("four");
        }
}
```

- ○ one
- ○ two
- ◉ three ✓
- ○ four
- ○ There is no output.

✓ What is the result? * 1/1

```
public static void main(String[] args) {
        String color = "Red";
        switch (color) {
        case "Red":
                System.out.println("Found Red");
        case "Blue":
                System.out.println("Found Blue");
        case "White":
                System.out.println("Found White");
                break;
        Default:
                System.out.println("Found Default");
        }
}
```

- ○ Found Red.
- ○ Found Red Found Blue.
- ◉ Found Red Found Blue Found White. ✓
- ○ Found Red Found Blue Found White Found Default.

✓ What is the result? * 1/1

```
class X {
        String str = "default";
        X(String s) { str = s; }
        void print() { System.out.println(str); }
        public static void main(String[] args) { new X("hello").print(); }
}
```

- (●) Hello ✓
- ( ) Default
- ( ) Compilation fails.
- ( ) The program prints nothing.
- ( ) An exception is thrown at run time.

✗ What is the result? * 0/1

```
import java.text.*;
public class Align {
        public static void main(String[] args) throws ParseException {
                String[] sa = {"111.234", "222.5678"};
                NumberFormat nf = NumberFormat.getInstance();
                nf.setMaximumFractionDigits(3);
                for (String s : sa) { System.out.println(nf.parse(s)); }
        }
}
```

- (●) 111.234 222.567 ✗
- ( ) 111.234 222.568
- ( ) 111.234 222.5678
- ( ) An exception is thrown at runtime.

Respuesta correcta

- (●) 111.234 222.5678

✗ What is the result? * 0/1

```
11. class Person {
12. String name  = "No name";
13. public Person (String nm) { name = nm}
14. }
15.
16. class Employee extends Person {
17.  String empID = "0000";
18.  public Employee (String id) { empID "
id; }
19. }
20.
21. public class EmployeeTest {
22.  public static void main(String[] args)
{
23.   Employee e = new Employee("4321");
24.   System.out.println(e.empID);
25.  }
26. }
```

- (●) 4321. ✗
- ( ) 0000.
- ( ) An exception is thrown at runtime.
- ( ) Compilation fails because of an error in line 18.

Respuesta correcta

- (●) Compilation fails because of an error in line 18.

✗ Which code fragment is illegal? * 0/1

- (●) Class Base1 { abstract class Abs1 { } } ✗
- ( ) Abstract class Abs2 { void doit() { } }
- ( ) class Base2 { abstract class Abs3 extends Base2 { } }
- ( ) class Base3 { abstract int var1 = 89; }

Una variable no puede indicarse como abstracta.

Respuesta correcta

- (●) class Base3 { abstract int var1 = 89; }

✗ In Java the difference between throws and throw Is: * 0/1

- (●) Throws throws an exception and throw indicates the type of exception that the method. ✗
- ( ) Throws is used in methods and throw in constructors.
- ( ) Throws indicates the type of exception that the method does not handle and throw an exception.

Respuesta correcta

- (●) Throws indicates the type of exception that the method does not handle and throw an exception.

✗ Which three methods, inserted individually at line, will correctly complete class Two(Choose three)? *0/1

```
10.  class One {
11.      void foo() { }
12.  }
13.  class Two extends One {
14.      //insert method here
15.  }
```

- [x] int foo() { /* more code here */ } ✓
- [ ] void foo() { /* more code here */ }
- [ ] public void foo() { /* more code here */ }
- [ ] private void foo() { /* more code here */ }
- [ ] protected void foo() { /* more code here */ }

Respuesta correcta

- [x] int foo() { /* more code here */ }
- [x] public void foo() { /* more code here */ }
- [x] protected void foo() { /* more code here */ }

✗ What is the result? * 0/1

```
public class MyStuff {
        String name;
        MyStuff (String n) { name = n; }
        public static void main (String[] args) {
                MyStuff m1 = new MyStuff ("guitar");
                MyStuff m2 = new MyStuff ("tv");
                System.out.println (m2.equals(m1));
        }
        public boolean equals (Object o) {
                MyStuff m = (MyStuff) o;
                if (m.name != null) { return true; }
                return false;
        }
}
```

- (●) The output is true and MyStuff fulfills the Object.equals() contract ✗
- ( ) The output is false and MyStuff fulfills the Object.equals() contract
- ( ) The output is true and MyStuff does NOT fulfill the Object.equals() contract.
- ( ) The output is false and MyStuff does NOT fulfill the Object.equals() contract

Respuesta correcta

- (●) The output is true and MyStuff does NOT fulfill the Object.equals() contract.

✗ Which two statements are true? * 0/1

- [x] An abstract class can implement an interface. ✓
- [ ] An abstract class can be extended by an interface.
- [ ] An interface CANNOT be extended by another interface.
- [ ] An interface can be extended by an abstract class.
- [ ] An abstract class can be extended by a concrete class.
- [ ] An abstract class CANNOT be extended by an abstract class

Respuesta correcta

- [x] An abstract class can implement an interface.
- [x] An abstract class can be extended by a concrete class.

✗ Which three lines will compile and output "Right on!"? * 0/1

```
13.  public class Speak {
14.      public static void main(String[] args) {
15.          Speak speakIT = new Tell();
16.          Tell tellIt = new Tell();
17.          speakIT.tellItLikeItIs();
18.          (Truth) speakIT.tellItLikeItIs();
19.          ((Truth) speakIT).tellItLikeItIs();
20.          tellIt.tellItLikeItIs();
21.          (Truth) tellIt.tellItLikeItIs();
22.          ((Truth) tellIt).tellItLikeItIs();
23.      }
24.  }

class Tell extends Speak implements Truth {
    @Override
    public void tellItLikeItIs() {
        System.out.println("Right on!");
    }
}

interface Truth {
    public void tellItLikeItIs();
}
```

- [x] Line 17 ✗
- [ ] Line 18
- [ ] Line 19
- [ ] Line 20
- [ ] Line 21
- [ ] Line 22

Respuesta correcta

- [x] Line 19
- [x] Line 20
- [x] Line 22

✗ What is the result? * 0/1

```
public static void main(String[] args) {
        System.out.println("Result: " + 2 + 3 + 5);
        System.out.println("Result: " + 2 + 3 * 5);
}
```

- (●) Result: 10 Result: 30 ✗
- ( ) Result: 25 Result: 10
- ( ) Result: 235 Result: 215
- ( ) Result: 215 Result: 215
- ( ) Compilation fails.

Respuesta correcta

- (●) Result: 235 Result: 215

✗ Which three options correctly describe the relationship between the classes? *0/1

```
class Class1 { String v1; }
class Class2 {
        Class1 c1;
        String v2;
}
class Class3 { Class2 c1; String v3; }
```

- [x] Class2 has-a v3. ✗
- [ ] Class1 has-a v2.
- [ ] Class2 has-a v2.
- [ ] Class3 has-a v1.
- [ ] Class2 has-a Class3.
- [ ] Class2 has-a Class1.

Respuesta correcta

- [x] Class2 has-a v2.
- [x] Class3 has-a v1.
- [x] Class2 has-a Class1.

✗ Which is true? * 0/1

```
5.  class Building { }
6.      public class Barn extends Building {
7.          public static void main(String[] args) {
8.              Building build1 = new Building();
9.              Barn barn1 = new Barn();
10.             Barn barn2 = (Barn) build1;
11.             Object obj1 = (Object) build1;
12.             String str1 = (String) build1;
13.             Building build2 = (Building) barn1;
14.         }
15.     }
```

Which is true?

- (●) If line 10 is removed, the compilation succeeds. ✗
- ( ) If line 11 is removed, the compilation succeeds.
- ( ) If line 12 is removed, the compilation succeeds.
- ( ) If line 13 is removed, the compilation succeeds.
- ( ) More than one line must be removed for compilation to succeed.

Respuesta correcta

- (●) If line 12 is removed, the compilation succeeds.

✓ What is the DTO pattern used for? * 1/1

- (●) To implement the data access layer ✓
- ( ) To exchange data between processes
- ( ) To implement the presentation layer.

✗ What is the result? * 0/1

```
class Alpha { String getType() { return "alpha"; } }
class Beta extends Alpha { String getType() { return "beta"; } }
public class Gamma extends Beta { String getType() { return "gamma"; }
        public static void main(String[] args) {
                Gamma g1 = new Alpha();
                Gamma g2 = new Beta();
                System.out.println(g1.getType() + " " + g2.getType());
        }
}
```

- (●) Alpha beta ✗
- ( ) Beta beta.
- ( ) Gamma gamma.
- ( ) Compilation fails.

Respuesta correcta

- (●) Compilation fails.

✗ The SINGLETON pattern allows: * 0/1

- (●) Have a single instance of a class and this instance cannot be used by other classes ✗
- ( ) Having a single instance of a class, while allowing all classes have access to that instance.
- ( ) Having a single instance of a class that can only be accessed by the first method that calls it.

Respuesta correcta

- (●) Having a single instance of a class, while allowing all classes have access to that instance.

✗ What is the result? * 0/1

```
import java.util.*;
public class MyScan {
        public static void main(String[] args) {
                String in = "1 a 10 . 100 1000";
                Scanner s = new Scanner(in);
                int accum = 0;
                for (int x = 0; x < 4; x++) {
                        accum += s.nextInt();
                }
                System.out.println(accum);
        }
}
```

- (●) 11 ✗
- ( ) 111
- ( ) 1111
- ( ) An exception is thrown at runtime.

Respuesta correcta

- (●) An exception is thrown at runtime.

✗ Which statement, when inserted into line " // TODO code application logic here", is valid in compilation time change? *0/1

```
public class SampleClass {
        public static void main(String[] args) {
                AnotherSampleClass asc = new AnotherSampleClass();
                SampleClass sc = new SampleClass();
                // TODO code application logic here
        }
}
class AnotherSampleClass extends SampleClass { }
```

- (●) asc = sc; ✗
- ( ) sc = asc;
- ( ) asc = (Object) sc;
- ( ) asc= sc.clone();

Respuesta correcta

- (●) sc = asc;

✗ What changes will make this code compile? * 0/1

```
class X {
        X() { }
        private void one() { }
}
public class Y extends X {
        Y() { }
        private void two() {
        one();
        }
        public static void main(String[] args) {
        new Y().two();
        }
}
```

- (●) Adding the public modifier to the declaration of class X. (✗
- ( ) Adding the protected modifier to the X() constructor.
- ( ) Changing the private modifier on the declarationof the one() method to protected.
- ( ) Removing the Y() constructor
- ( ) Removing the private modifier from the two() method.

Respuesta correcta

- (●) Changing the private modifier on the declarationof the one() method to protected.

✗ What is the result? * 0/1

```
try {
        // assume "conn" is a valid Connection object
        // assume a valid Statement object is created
        // assume rollback invocations will be valid
        // use SQL to add 10 to a checking account
        Savepoint s1 = conn.setSavePoint();
        // use SQL to add 100 to the same checking account
        Savepoint s2 = conn.setSavePoint();
        // use SQL to add 1000 to the same checking account
        // insert valid rollback method invocation here
} catch (Exception e) { }
```

- [x] If conn.rollback(s1) is inserted, account will be incremented by 10. ✓
- [ ] If conn.rollback(s1) is inserted, account will be incremented by 1010.
- [ ] If conn.rollback(s2) is inserted, account will be incremented by 100
- [ ] If conn.rollback(s2) is inserted, account will be incremented by 110.
- [ ] If conn.rollback(s2) is inserted, account will be incremented by 1110

Respuesta correcta

- [x] If conn.rollback(s1) is inserted, account will be incremented by 10.
- [x] If conn.rollback(s2) is inserted, account will be incremented by 110.

✗ What is the result? * 0/1

```
class MyKeys {
        Integer key;
        MyKeys(Integer k) { key = k; }
        public boolean equals(Object o) {
                return ((MyKeys) o).key == this.key;
        }
}
```

And this code snippet:

```
Map m = new HashMap();
MyKeys m1 = new MyKeys(1);
MyKeys m2 = new MyKeys(2);
MyKeys m3 = new MyKeys(1);
MyKeys m4 = new MyKeys(new Integer(2));
m.put(m1, "car");
m.put(m2, "boat");
m.put(m3, "plane");
m.put(m4, "bus");
System.out.print(m.size());
```

- (●) 2 ✗
- ( ) 3
- ( ) 4
- ( ) Compilation fails.

Respuesta correcta

- (●) 4

✗ Which two declarations will compile? * 0/1

```
14.    public static void main(String[] args) {
15.            Int a, b, c = 0;
16.            int a, b, c;
17.            int g, int h, int i = 0;
18.            int d, e, f;
19.            Int k, l, m, = 0;
20.    }
```

- [x] Line 15. ✗
- [ ] Line 16.
- [ ] Line 17.
- [ ] Line 18.
- [ ] Line 19.
- [ ] Line 20.

Respuesta correcta

- [x] Line 16.
- [x] Line 18.

✗ Which three implementations are valid? * 0/1

```
interface SampleCloseable {
        public void close() throws java.io.IOException;
}
```

- [x] class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something } } ✓
- [ ] class Test implements SampleCloseable { public void close() throws Exception { // do something } }
- [ ] class Test implements SampleCloseable { public void close() throws FileNotFoundException { // do something } }
- [ ] class Test extends SampleCloseable { public void close() throws java.io.IOException { // do something } }
- [ ] class Test implements SampleCloseable { public void close() { // do something }}

Respuesta correcta

- [x] class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something } }
- [x] class Test implements SampleCloseable { public void close() throws FileNotFoundException { // do something } }
- [x] class Test implements SampleCloseable { public void close() { // do something }}

✗ Which one is valid as a replacement for foo? * 0/1

```
public static void main(String[] args) {
        Boolean b1 = true;
        Boolean b2 = false;
        int i = 0;
        while (foo) { }
}
```

- (●) b1.compareTo(b2) ✗
- ( ) i = 1
- ( ) i == 2? -1:0
- ( ) foo.equals("bar")

Respuesta correcta

- (●) foo.equals("bar")

✗ What is the result? * 0/1

```
import java.util.*;
public class App {
        public static void main(String[] args) {
                List p = new ArrayList();
                p.add(7);
                p.add(1);
                p.add(5);
                p.add(1);
                p.remove(1);
                System.out.println(p);
        }
}
```

- (●) [7, 1, 5, 1] ✗
- ( ) [7, 5, 1]
- ( ) [7, 5]
- ( ) [7, 1]

Respuesta correcta

- (●) [7, 5, 1]

✗ What is the result? * 0/1

Given

```
public class SuperTest {
        public static void main(String[] args) {
                //statement1
                //statement2
                //statement3
        }
}
class Shape {
        public Shape() {
                System.out.println("Shape: constructor");
        }
        public void foo() {
                System.out.println("Shape: foo");
        }
}
class Square extends Shape {
        public Square() {
                super();
        }
        public Square(String label) {
                System.out.println("Square: constructor");
         }
        public void foo() {
                super.foo();
        }
        public void foo(String label) {
                System.out.println("Square: foo");
        }
}
```

What should statement1, statement2, and statement3, be respectively, in order to produce the result?

Shape: constructor
Shape: foo
Square: foo

- (●) Square square = new Square ("bar"); square.foo ("bar"); square.foo(); ✗
- ( ) Square square = new Square ("bar"); square.foo ("bar"); square.foo ("bar");
- ( ) Square square = new Square (); square.foo (); square.foo(bar);
- ( ) Square square = new Square (); square.foo (); square.foo("bar");
- ( ) Square square = new Square (); square.foo (); square.foo ();

Respuesta correcta

- (●) Square square = new Square (); square.foo (); square.foo("bar");

✗ What is the result? * 0/1

```
public static void main(String[] args) {
        int [][] array2D = { {0, 1, 2}, {3, 4, 5, 6} };
        System.out.print(array2D[0].length + "" );
        System.out.print(array2D[1].getClass().isArray() + "" );
        System.out.println(array2D[0][1]);
}
```

- (●) 3false1 ✗
- ( ) 2true3
- ( ) 2false3
- ( ) 3true1
- ( ) 3false3
- ( ) 2true1
- ( ) 2false1

Respuesta correcta

- (●) 3true1

✗ What is the result? * 0/1

```
public class SampleClass {
        public static void main(String[] args) {
                AnotherSampleClass asc = new AnotherSampleClass();
                SampleClass sc = new SampleClass();
                sc = asc;
                System.out.println("sc: " + sc.getClass());
                System.out.println("asc: " + asc.getClass());
        }
}
class AnotherSampleClass extends SampleClass { }
```

- (●) sc: class.Object asc: class.AnotherSampleClass ✗
- ( ) sc: class.SampleClass asc: class.AnotherSampleClass
- ( ) sc: class.AnotherSampleClass asc: class.SampleClass
- ( ) sc: class.AnotherSampleClass asc: class.AnotherSampleClass

Respuesta correcta

- (●) sc: class.AnotherSampleClass asc: class.AnotherSampleClass

✓ In the Java collections framework a Set is: * 1/1

- (●) A collection that cannot contain duplicate elements. ✓
- ( ) An ordered collection that can contain duplicate elements
- ( ) An object that maps value key sets and cannot contain values Duplicates

✗ What will make this code compile and run? * 0/1

```
01. public class Simple {
02.
03.     public float price;
04.     public static void main(String[] args) {
05.
06.         Simple price = new Simple();
07.         price = 4;
08.     }
09. }
```

- (●) Change line 3 to the following: public int price; ✗
- ( ) Change line 7 to the following: int price = new Simple();
- ( ) Change line 7 to the following: float price = new Simple ();
- ( ) Change line 7 to the following: price = 4f;
- ( ) Change line 7 to the following: price.price = 4;

Respuesta correcta

- (●) Change line 7 to the following: price.price = 4;

✗ What is the result? * 0/1

```
interface Rideable {
        String getGait();
}

public class Camel implements Rideable {
        int weight = 2;
        String getGait() {
                return " mph, lope";
        }
        void go(int speed) {
                ++speed;
                Weight++;
                int walkrate = speed * weight;
                System.out.print(walkrate + getGait());
        }
        public static void main(String[] args) {
                new Camel().go(8);
        }
}
```

- (●) 16 mph, lope ✗
- ( ) 24 mph, lope.
- ( ) 27 mph, lope.
- ( ) Compilation fails

Respuesta correcta

- (●) Compilation fails

✗ What is the result? * 0/1

```
class Atom {
        Atom() { System.out.print("atom "); }
}
class Rock extends Atom {
        Rock(String type) { System.out.print(type); }
}
public class Mountain extends Rock {
        Mountain() {
                super("granite ");
                new Rock("granite ");
        }
        public static void main(String[] a)  { new Mountain(); }
}
```

- (●) Compilation fails. ✗
- ( ) Atom granite.
- ( ) Granite granite.
- ( ) Atom granite granite.
- ( ) An exception is thrown at runtime.
- ( ) Atom granite atom granite.

Respuesta correcta

- (●) Atom granite atom granite.

✗ Which three are valid? (Choose three) * 0/1

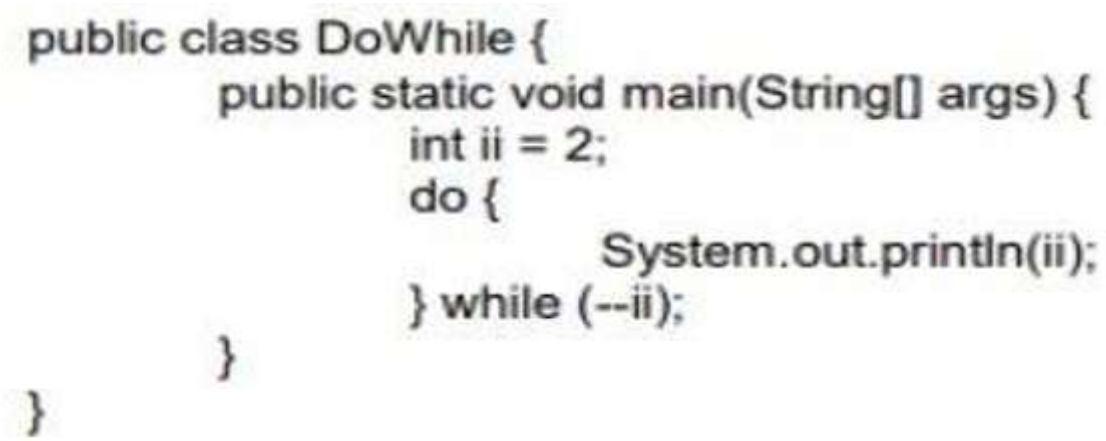
```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}
And:
ClassA p0 = new ClassA();
ClassB p1 = new ClassB();
ClassC p2 = new ClassC();
ClassA p3 = new ClassB();
ClassA p4 = new ClassC();
```

- [x] p0 = p1; ✓
- [ ] p1 = p2;
- [ ] p2 = p4;
- [ ] p2 = (ClassC)p1;
- [ ] p1 = (ClassB)p3;
- [ ] p2 = (ClassC)p4;

Respuesta correcta

- [x] p0 = p1;
- [x] p1 = (ClassB)p3;
- [x] p2 = (ClassC)p4;

✗ What is the result? * 0/1



```
public class DoWhile {
        public static void main(String[] args) {
                int ii = 2;
                do {
                        System.out.println(ii);
                } while (--ii);
        }
}
```

- (●) 2 1 2 ✗
- ( ) 1 0
- ( ) null
- ( ) An infinite loop.
- ( ) Compilation fails

Respuesta correcta

- (●) Compilation fails

✗ What is the result if the integer value is 33? * 0/1

```
public static void main(String[] args) {
        if (value >= 0) {
                if (value != 0) {
                        System.out.print("the ");
                } else {
                        System.out.print("quick ");
                }
                if (value < 10) {
                        System.out.print("brown ");
                }
                if (value > 30) {
                        System.out.print("fox ");
                } else if (value < 50) {
                        System.out.print("jumps ");
                } else if (value < 10) {
                        System.out.print("over ");
                } else {
                        System.out.print("the ");
                }
                if (value > 10) {
                        System.out.print("lazy ");
                } else {
                        System.out.print("dog ");
                }
                System.out.print("... ");
        }
}
```

- (●) The fox jump lazy ? ✗
- ( ) The fox lazy ?
- ( ) Quick fox over lazy ?

Respuesta correcta

- (●) The fox lazy ?

✗ What is the result? * 0/1

```
public class X {
        public static void main(String[] args) {
                String theString = "Hello World";
                System.out.println(theString.charAt(11));
        }
}
```

- (●) There is no output. ✗
- ( ) d is output.
- ( ) A StringIndexOutOfBoundsException is thrown at runtime.
- ( ) An ArrayIndexOutOfBoundsException is thrown at runtime.
- ( ) A NullPointException is thrown at runtime.
- ( ) A StringArrayIndexOutOfBoundsException is thrown at runtime.

Respuesta correcta

- (●) A StringIndexOutOfBoundsException is thrown at runtime.