Preguntas del documento Java_cuestionario1.pdf

1.- Excepciones de java permitidas?

A) org.springframework.Web.client.RestClientException
B) No conozco la pregunta
C) java.lang.NumberFormatException
D) com.bbva.apx.exception.db.NoResultException
E) com.bbva.elara.utility.interbackend.cics.exceptions.BusinessException

Resultado, c, esto debido a que es una de runtime, supongo el resto no lo son?

2.- Which three are bad practices?

a) checking for ArrayIndexOutOfBoundsException and ensuring that the program can recover if one occurs
b) checking for FileNotFoundException to inform a user that a filename entered is not valid
c) checking for error and, if necessary, restarting the program to ensure that users are unaware of problems
d) checking for ArrayIndexOutOfBoundsException when iterating through an array to determine when all elements have been visited.

Resultado, a, c y d, a porque no deberia siquiera suceder uno si es que esta bien hecha la logica del programa, c porque no se debe reiniciar el programa a lo loco en prod y d porque no se debe usar las excepciones en esa forma

3.- Indica a JUnit que la propiedad que usa esta anotacion es una simulacion y, por lo tanto, se inicializa cómo tal y es susceptible de ser inyectada por @InjectMocks

a) Mockito
b) Mock
c) Inject
d) InjectMock

Respuesta, Mock, no hay mucho que pueda añadir, el nombre es autoexplicativo

4.- Given

```
public static void main (String[] args){
        int[][] array 2d= {{0,1,2}, {3,4,5,6}};
        System.out.print(array2d[0].length+"");
        System.out.print(array2d[1].getClass().isArray()+"");
        System.out.print(array2d[0][1]);
}
```

Cual es el resultado?

a) 3false3
b) 3false1
c) 2false1
d) 3true1
e) 2true3

el resultado es d, la longitud del indice 0 es 3, es un array y 0,1 es 1

5.-Which two statements are true?

a) An interface CANNOT be extended by another interface
b) An abstract class can be extended by a concrete class

c) An abstract class CANNOT be extended by an abstract class. An interface can be extended by an abstract class.
d) An abstract class can implement an interface.
e) An abstract class can be extended by an interface.

Respuestas, b y d, de las opciones son las que si se pueden hacer

6.- Which five methods, inserted independently at line 5 will compile?

```
public class Blip{
        protected int blipvert(int x){return 0;}
}
class Vert extends Blip{
        //insert code here
}
```

a) private int blipvert(long x){return 0;}
b) protected int blipvert(long x){return 0;}
c) protected long blipvert(int x, int y){return 0;}
d) public int blipvert(int x){return 0;}
e) private int blipvert(int x){return 0;}
f) protected long blipvert(int x){return 0;}
g) protected long blipvert(long x){return 0;}

respuestas, a, b, c, d y g, a, b, c y d porque son overload, no tienen la misma firma que el método original asi que se les permite existir, d, porque es un override con un modificador de alcance igual o mas amplio que el original.

7.- Given:

```
class Super{
        private int a;
        protected Super(int a){this.a=a;}
}
…
class Sub extends Super{
        public Sub(int a){super(a);}
        public Sub(){this.a=5;}
}
```

Which two independently, will allow Sub to compile? choose two

a) change line 2 to: public int a;
b) change line 13 to: public Sub(){super(5);}
c) change line 2 to: protected int a;
d) change line 13 to public Sub(){this(5);}
e) change line 13 to: public Sub(){super(a)

respuestas, b y c, el problema radica en que this.a=5; es la linea problematica, al cambiarlo para que mande llamar al constructor de la clase padre para que haga el cambio deberia hacer que funcione, en este caso con pasarlo al constructor super o al otro constructor de sub que acepte enteros deberia bastar.

8.-What is true about the class Wow?

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

```
        public void wow(){}
        private void wowza(){}
}
```

a) it compiles without error.
b) it does not compile because an abstract class cannot have private methods
c) it does not compile because an abstract class cannot have instance variables
d) it does not compile because an abstract class must have at least one abstract method
e) it does not compile because an abstract class must have a constructor with no arguments

resultado, a, ninguna de las cosas que dicen sobre las clases abstractas es cierto, ademas de que no hay problemas de sintaxis o modificadores en el código

9.- What is the result?

```
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();}
}
```

a) compilation fails
b) atom granite
c) granite granite
d) atom granite granite
e) an exception is thrown at runtime
f) atom granite atom granite

resultado, f, al momento de instanciar una clase, se llama a su constructor y este llama al constructor padre primero, el cual es resuelto primero imprimiendo atom para luego imprimir granite de rock, y lo hace una vez mas por el new Rock

10.-What is printed out when the program is executed?

```
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”);
}
```

a) one
b) two
c) three
d) four
e) there is no output

respuesta, c, esto porque tiene la firma del método main que se busca para ejecutar el programa

11.- What is the result?

```
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();
        }
}
```

a) cougar c f
b) feline cougar c f
c) feline cougar c c
d) compilation fails

es c, al momento de cambiarle el valor a type toma el de la super clase a cambiar

12.- What is the result?

```
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());
        }
}
```

a) alpha beta
b) beta beta

c) gamma gamma
d) compilation fails

es d, no se puede meter en un hijo a un padre

13.- What is the result?

```
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);
        }
}
```

a) 11
b) 11
c) 1111
d) an exception is thrown at runtime

d, esto debido a que una vez llega a la 'a' manda el error de que el token no es entero

14.-

```
public class Bees{
        public static void main(String[] argos){
                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");
        }
}
```

a) the program prints 1 then 2 after 5 seconds
b) the program prints 1 thrown to main
c) the program prints 1 2 thrown to main
d) the program prints 1 then t1 waits for its notification

la respuesta es b, manda IllegalMonitorStateException por una cuestion de hilos

15.-Which statement is true?

```
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[] argos){
                        ExtendedA ext = new ExtendedA(420);
                        System.out.print(ext.numberOfInstances);
                }
}
```

a) 420 is the output
b) an exception is thrown at runtime
c) all constructors must be declared public
d) constructors cannot use the private modifier
e) constructors cannot use the protected modifier

a es la respuesta correcta, esto porque todo estaba bien

16.- the singleton pattern allows:

a) to have a single instance of a class and this instance cannot be used by other classes
b) having a single instance of a class, while allowing all classes to have access to that instance
c) having a single instance of a class that can only be accessed by the first method that calls it.

respuesta es b

17.- What is the result?

```
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));}
        }
}
```

a) 111.234 222.567
b) 111.234 222.568
c) 111.234 222.5678
d) an exception is thrown at runtime

respuesta, d, number format solo se encuentra parseando, no dándole formato para que respete el maximum fraction digits

18.-What is the result?

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

A) Square square = new Square(“bar”); square.foo(“bar”); square.foo();
B) Square square = new Square(“bar”); square.foo(“bar”); square.foo(“bar”);
C) Square square = new Square(); square.foo(); square.foo(bar);
D) Square square = new Square(); square.foo(); square.foo(“bar”);
E) Square square = new Square(); square.foo(); square.foo();

la respuesta es d, el constructor default manda llamar super que es el que tiene el valor necesario para imprimir, c tiene una variable sin declarar siendo pasada lo cual causaria error de compilacion mientras que la d es la que manda pasar un string que es el método que se necesitaba

19.- Which three implementations are valid?

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

a) class TestImplements SampleCloseable{public void close()throws java.io.IOException{//do something}}
b) class TestImplements SampleCloseable{public void close() throws Exception{//do something}}
c) class TestImplements SampleCloseable{public void close() throws FileNotFoundException{//do something}}
d) class Test extends SampleCloseable{public void close() throws java.io.IOException{//do something}}
e) class TestImplements SampleCloseable{public void close(){//do something}}

respuestas, a c y e, b no es una opcion ya que lanza una excepcion mas general, no se puede hacer eso ya que tiene que ser una excepcion dentro del mismo tipo para que sea valida la implementación de la interfaz, d tampoco es porque una interfaz no se extiende, se implementa, solo se puede extender entre interfaces

20.-What is the result?

```
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());
```

a) 2
b) 3
c) 4
d) compilation fails

respuesta 4, al ser Integer y no enteros son objetos codependientes entre si, por eso al usar el operador == siempre sera falso a menos que uno de ellos llegue a apuntar al mismo objeto

21.- What value of x,y,z will produce the following result?
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("----");
        }
}
```

a) int x=4, y=3, z=2;
b) int x=3, y=2, z=3;
c) int x=2, y=3, z=3;
d) int x=2, y=3, z=4;
e) int x=4, y=2, z=3;

respuesta d, es la unica que tiene z=4 que es la condicion para que se imprima el 1234

22.- Which three lines will compile and output "right on!"?

```
public class Speak{
        public static void main(String[] args){
                Speak speakIT=new Tell();
                Tell tellIT = new TelII(); //16
                speakIT.tellItLikeItIs(); //17
                (Truth) speakIT.tellItLikeItIs(); //18
                ((Truth)speakIT).tellItLikeItIs(); //19
                tellItl.tellItLikeItIs(); //20
                (Truth)tellIt.tellItLikeItIs(); //21
                ((Truth)tellIt).tellItLikeItIs(); //22
        }
}
class Tell extends Speak implements Truth{
        @Override
        public void tellItLikeItIs(){
                System.out.println("Right on!");
        }
}
interface Truth{
        public void tellItLikeItIs();
}
```

a) line 17
b) line 18
c) line 19
d) line 20
e) line 21
f) line 22

respuesta c, d y f, a no es porque porque speak no tiene un método llamado asi, causando un error, b no es porque el casteo se hace a todo el resultante de lo que llama el método, e tampoco es porque hace esto mismo

23.- Cual es el resultado?

```
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);
	}
}
```

a) [7,5]
b) [7,1]
c) [7,5,1]
d) [7,1,5,1]

la respuesta es c, ya que remove quita el primer elemento que concuerde con el valor pasado

24.-What is the result?

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

a) 22
b) 12
c) 32
d) 33

respuesta, a, con el primer decremento es 3, al momento de hacerle el pre decremento se vuelve 2 el cual es impreso, para luego imprimir de nuevo el valor

25.-In java the difference between throws and throw is:

a) throws throws an exception and throw indicates the type of exception that the method throws
b) Throws is used in methods and throw in constructors
c) throws indicates the type of exception that the method does not handle and throw an exception

respuesta, c

26.-Which statement, when inserted into line “//TODO code application logic here”, is valid in compilation time change?

```
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{}
```

a) asc=sc;
b) sc=asc;
c) asc=(Object)sc;
d) asc= sc.clone();

respuesta b, ya que es el hijo entrando en una padre, lo cual es valido

27.- What is the result?

```
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]);
        }
}
```

a) 4 Null
b) Null 4
c) An illegalArgumentException is thrown at run time
d) 4 An ArrayIndexOutOfBoundsException is thrown at run time

respuesta, d, esto porque el elemento 1 solo tiene dos elementos dentro, se intenta acceder a uno al que no existe causando el error

28.- Which three are valid?

```
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();
```

a) p0=p1;
b) p1=p2;
c) p2=p4;
d) p2=(ClassC)p1;
e) p1=(ClassB)p3;
f) p2=(ClassC)p4

resultados, a, e y f, a porque se mete al hijo en una clase padre, e porque se castea una clase b que entro en a para meterla a una b y f porque se hace lo mismo que con e pero con la clase c en vez de la b

29.- Which three options correctly describe the relationship between the classes?

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

a) Class2 has a v3
b) Class1 has a v2
c) Class2 has a v2
d) Class3 has a v1
e) Class2 has a Class3
f) Class2 has a Class1

respuestas, c,d y f, c porque es de las que estan declaradas, d porque tiene una v1 por class 2 que tiene una class 1, f porque tambien esta declarada en sus atributos

30.- What is the result?

```
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+" ");}
```

a) 2 3 5 7
b) 2 7 5 3
c) 7 5 3 2
d) compilation fails

respuesta, c, esto porque el MySort lo unico que hace es invertir el orden de comparación, en vez de menor a mayor va de mayor a menor

31.-Which two possible outputs?

```
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");
        }
```

}

A) before if clause exception in thread “main” java.lang.exception at main.doSomething (main.java:21) at Main.main (main.java:15)
B) before if clauseException in thread “main” java.lang.exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15) after if clause
C) Exception in thread “main” java.lang.exception at Main.doSomething (Main.java:21) at main.main (Main.java:15)
D) before if clause after if clause

respuesta, a y d, a porque imprime before if clause antes de enviar la exception, una vez enviada se detiene el resto del programa, mientras que d es el flujo normal del programa si no existiera el if

32.- What is the result?

```
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 “red”:
                        System.out.println(“found white”);
                        break;
                default:
                        System.out.println(“found default”);
        }
}
```

a) found red
b) found red found blue
c) found red found blue found white
d) found red found blue found white found default

respuesta, c, esto porque al entrar a red no hay un break que lo detenga

33.- What is the result?

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

a) 7
b) 12
c) 19
d) compilation fails
e) an exception is thrown at run time

resultado, b, esto porque solo lo modifica dentro del método, al ser una copia local de la variable no hace que se quede este cambio al momento de acabar con el método

34.- Which is true?

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

A) if line 10 is removed, the compilation succeeds
B) if line 11 is removed, the compilation succeeds
C) if line 12 is removed, the compilation succeeds
D) if line 13 is removed, the compilation succeeds
E) More than one line must be removed for compilation to succeed

resultados, c, no se puede castear a otra clase que no tiene relacion alguna con el objeto original, en este caso string con building

35.- What is the result if the integer value is 33?

```
public static void main(String[] args){
    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(“... “);
```

}

a) the fox jump lazy …
b) the fox lazy …
c) quick fox over lazy …
d) quick fox the …

resultado, b, esto porque al evaluar los ifs, entra al primero e imprime the, no entra al segundo, entra al 3ero imprimiendo fox y al final entra imprimiendo lazy

36.- What is the result?

```
class Person{
        String name= “no name”;
        public Person(String nm){name=nm;}
}
class Employee extends Person{
        String empID=”0000”;
        public Employee(String id){
                empID=””;
        }
}
public class EmployeeTest{
        public static void main(String[] args){
                Employee e=new Employee(“4321”);
                System.out.println(e.empID);
        }
}
```

a) 4321
b) 0000
c) an exception is thrown at runtime
d) compilation fails because of an error in line 18

respuesta, d, esto porque falta un super

37.- Which code fragment is illegal?

a) class Base1{abstract class Abs1{}}
b) abstract class Abs2{void doit(){}}
c) class Base2{abstract class abs3 extends base 2{}}
d) class base3 {abstract int var1=89;}

respuesta d, no se debe declarar valores de variables abstractas

38.-What is the result?

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

a) result 10 result 30
b) result 25 result 10
c) result 235 result 215
d) result 215 result 215
e) compilation fails

resultado, c, esto debido a que se suma cómo si fuera string en la primera, en la segunda por orden de operacion primero se multiplica 3 y 5, dando el resultado sumado a las cadenas cómo cadena

39.- 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;
        }
}
```
a) the output is true and myStuff fulfills the object.equals() contract
b) the output is false and mystuff fulfills the object.equals() contract
c) the output is true and MyStuff does not fulfill the object.equals() contract
d) the output is false and mystuff does not fulfill the object.equals() contract

respuesta, c, devuelve true debido a que el nombre es diferente de null, pero esto mismo hace que sea igual con todo objeto que tenga asignado un nombre que no sea null

40.-Which one is valid as a replacement for foo?
```
public static void main(String[] args){
        Boolean b1=true;
        Boolean b2= false;
        int i=0;
        while(foo){}
}
```
a) b1.compareTo(b2)
b) i=1
c) i==2?-1:0
d) foo.equals(“bar”)

respuesta, d

41.-what is the result?
```
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);
        }
}
```

a) 16 mph, lope
b) 24 mph, lope
c) compilation fails
d) 27 mph, lope

resultado, c, falla la compilacion debido a que getGait de Camel no es publica, a diferencia de la interfaz que lo establece

42.- What is the result

```
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():}
}
```

a) hello
b) default
c) compilation fails
d) the program prints nothing
e) an exception is thrown at run time

resultado, a, el programa no tiene nada de malo y se ejecuta de forma correcta

43.- what is the result?

```
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]);
}
```

a) 14
b) 15
c) 24
d) 25
e) 34
f) 35

respuesta, es f, arraycopy toma el array de referencia, desde el punto que se le da, copiando hacia el array que se le da, desde el punto de referencia que se le da y con la longitud que se le da, en este caso el array resultante es 1,3,4,4,5

44.- what is the result?

```
public class Main{
	public static void main(String[] args){
		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;

		System.out.println(“el valor de w es: “+w);
	}
}
```

a) 0
b) 30
c) 60

resultado, a, no entra a ninguno de los ifs y por ende no se altera el valor de z

45.- what is the result?

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

a) 2 1 2
b) 1 0
c) null
d) an infinite loop
e) compilation fails

resultado, e, while necesita un booleano, no acepta ese tipo de expresiones

46.- what changes will make this code compile?

```
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();
        }
}
```

a) adding the public modifier to the declaration of class X
b) removing the y() constructor
c) removing the private modifier from the two() method
d) adding the protected modifier to the x() constructor
e) changing the private modifier on the declaration of the one() method to protected

resultado, e, esto porque el private del método one de la clase x hace que sea inalcanzable por y al momento de intentar ejecutarlo

47.- which two declarations will compile?

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

a) line 15
b) line 16
c) line 17
d) line 18
e) line 19
f) line 20

resultado, a, b, d, e, esto porque son declaraciones válidas

48.- Which three methods, inserted individually at line, will correctly complete class Two? choose three

```
class One{
        void foo(){}
}
class Two extends One{
        //insert method here
}
```

a) public void foo(){/*more code here*/}
b) private void foo(){/*more code here*/}
c) protected void foo(){/*more code here*/}
d) int foo(){/*more code here*/}

resultado, a,c, y d, porque son overrides con un scope mas amplio que el original

49.- what is the result?

```
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){}
```

a) if conn.rollback(s1) is inserted, account will be incremented by 10.
b) if conn.rollback(s1) is inserted, account will be incremented by 1010
c) if conn.rollback(s2) is inserted, account will be incremented by 100
d) if conn.rollback(s2) is inserted, account will be incremented by 110.
e) if conn.rollback(s2) is inserted, account will be incremented by 1110.

resultado, a y d, esto porque el rollback fue hecho despues de 10 en s1 y despues de 10 y 100 en s2.

50.- what is the result?

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

a) result 8 result 8
b) result 35 result 8
c) result 8 result 35
d) result 35 result 35

resultado b, suma en string cómo 35 en el primero, para luego sumar cómo entero y luego cómo string en el segundo

51.- What is the result?

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

a) there is no output
b) d is output
c) A StringIndexOutOfBoundsException is thrown at runtime
d) an ArrayIndexOutOfBoundsException is thrown at runtime
e) a NullPointException is thrown at runtime
f) a StringArrayIndexOutOfBoundsException is thrown at runtime

respuesta, c, no hay indice 11

52.- What will make this code compile and run?

```
public class Simple{
        public float price;
        public static void main(String[] args){
                Simple price=new Simple();
                price=4;
        }
}
```

a) change line 3 to the following: public int price

b) change line 7 to the following: int price= new simple();
c) change line 7 to the following: float price = new Simple();
d) change line 7 to the following: price=4f;
e) change line 7 to the following:price.price=4;

respuesta, a y e, al cambiarle a int se podra aceptar el valor dado, pero la importante es la e, al cambiar a price.price se podra acceder al price que se tiene interno de simple, ya que si se hace asi manda error por tipos incompatibles

53.- in the java collections framework a set is:

a) a collection that cannot contain duplicate elements
b) an ordered collection that can contain duplicate elements
c) an object that maps value key sets and cannot contain values duplicates

respuesta, a

54.- What is the result?

```
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{}
```

a) sc: class.object asc: class.AnotherSampleClass
b) sc: class.SampleClass asc: class.AnotherSampleClass
c) sc: class.AnotherSampleClass asc: class.SampleClass
d) sc: class.AnotherSampleClass asc: class.AnotherSampleClass

respuesta, d, ya que aunque sc sea una sample class, el padre puede albergar hijos, en este caso another sample class, mientras mantiene su tipo de clase

55.- which declaration initializes a boolean variable?

a) boolean j=(1<5);
b) boolean m=null;
c) boolean h=1;
d) boolean k=0;

respuesta, a, es la unica que tiene una expresion booleana valida para ser evaluada y cuyo resultado pueda ser pasado cómo booleano

56.- which two will compile and can be run successfully using the following command?
Java Fred1 Hello walls

a) abstract class Fred1{public static void main(String[]args){System.out.println(args[2]);}}
b) class Fred1{public static void main(String args){System.out.println(args[1]);}}
c) class Fred1{public static void main(String[] args){System.out.println(args[1]);}}
d) class Fred1{public static void main(String[] args){System.out.println(args);}}

respuesta d, imprimiria la direccion de args

57.-how many times is 2 printed?

```
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;
                }
        }
}
```

a) thrice
b) it is not printed because compilation fails
c) twice
d) zero
e) once

respuesta, e, despues de la primera ejecución de while se rompe el ciclo

58.-which two are valid instantiations and initializations of multidimensional array?

a) array2D[0][0]=1; array2D[0][1]=2; array2D[1][0]=3; array2D[1][1]=4;
b) array3D[0][0]=array; array3D[0][1]=array; array3D[1][0]=array; array3D[0 [1]= array;
c) int[][] array2D={0,1};
d) int[][][] array3D={{0,1},{2,3},{4,5}};int[] array={0,1}; int[][][] array3D=newint[2][2][2];
e) int [][] array2d={{0,1,2,4}{5,6}}; int[][] array2D=new int[2][2];

respuesta, a y d, a muestra cómo meter variables a un array ya declarado de dimensiones dos, mientras que d muestra con un array de 3 dimensiones de dos diferentes formas y uno de una dimension

59.-the standard API for accessing databases in java is:

a) JPA/Hibernate
b) jdbc
c) odbc

respuesta, jdbc, es el estandar para acceder a database

60.- What is the best way to test that the values of h1 and h2 are the same?

```
public static void main(String[] args){
        String h1= “bob”;
        String h2= new String (“bob”);
}
```

a) if(h1.equals(h2))
b) if(h1==h2)
c) if(h1.toString()==h2.toString())
d) if(h1.same(h2))

respuesta, a, es la forma en que se puede comparar dos strings si alguna de ellas o ambas no se encuentran en el pool de strings.

61.- what is the result?

```
public class Boxer1{
        Integer i;
        int x;
        public Boxer1(int y){
                x=i+y;
                System.out.println(x);
        }
        public static void main(String[] args){
                new Boxer1(new Integer(4));
        }
}
```

a) NullPointerException occurs at runtime
b) the value “4” is printed at the command line
c) an IllegalStateException occurs at runtime
d) compilation fails because of an error in line 9
e) compilation fails because of an error in line 5
f) a NumberFormatException occurs at runtime

respuesta, a, NullPointerException sucede porque Integer i no anda inicializado

62.-given:
We have the following Java class:

```
package gal.dicoruna.example;
public class C{
        protected String v;
        …
}
```

a) from the class, the package, subclasses
b) any site but only read if it is outside the package
c) from the class, the package, subclasses and all sites

respuesta, a, ya que al ser protected su acceso es mas restrictivo pero no tanto cómo el default y private

63.- What is the result?

```
class Foo{
        public void addFive(){ a+=5; System.out.println(“F”);}
}
class Bar extends Foo{
        public int a=0;
        public void addFive(){this.a +=5; System.out.println(“B”);}
}
invoked with:
        Foo f=new Bar();
        f.addFive();
        System.out.println(f.a);
```

a) b 3
b) compilation fails
c) f 8
d) b 8
e) f 13
f) f 3
g) an exception is thrown at runtime b 13

respuesta, b, falla la compilacion porque no se tiene una a en la clase padre, asi que el método padre de addFive no sabe a que a se refiere con esa a, causando error de compilacion

64.- what is the result?

```
public class ScopeTest{
        int z;
        public static void main(String[] args){
                ScopeTest myScope new ScopeTest();
                int z=6;
                System.out.println(z);
                myScope.doStuff();
                System.out.print(z);
                System.out.print(myScope.z);
        }
        void doStuff(){
                int z=5;
                doStuff2();
                System.out.print(myScope.z);
        }
        void doStuff2(){
                z=4;
        }
}
```

a) 6554
b) 6564
c) 6565
d) compilation fails
e) 6566

resultado, d, falla al tener el myScope fuera de scope en la funcion doStuff

65.-What is the result?

```
public class Barn{
        public static void main(String[] args){
                new Barn().go("hi",1);
                new Barn().go("hi","world",2);
        }
        public void go(String… y, int x){
                System.out.print(y[y.length-1]+" ");
        }
}
```

a) worl world
b) hi hi
c) hi world
d) an exception is thrown at runtime
e) compilation fails

respuesta, e, falla al compilar debido a que el varargs debe estar al final de la declaracion de variables de método

66.- What is the result if you try to compile Truthy.java and then run it with assertionsenabled?

```
public class Truthy{
        public static void main(String[] args){
                int x=7;
                assert (x==6) ? “x==6” : “x != 6”;
        }
}
```

a) truthy.java compiles and the output is x!=6
b) truthy.kava compiles and an assertionerror is thrown with x!=6 as additionaloutput
c) truthy.java does not compile
d) truthy.java compiles and an assertionerror is thrown with no additional output

resultado, c, no compila ya que se requiere una unica condicion que se evalue para el assert

67.- What is the result? Given:
no code given

68.-what is the result when this program is executed?

```
public class Student{
        public String name=””;
        public int age=0;
        public String major=”undeclared”;
        public boolean fulltime=true;
        public void display(){
                System.out.println(“name:” + name + “major: “+major);
        }
        public boolean isFullTime(){
                        return fulltime;
                }
}
```

and:

```
class TestStudent{
        public static void main(String[] args){
                Student bob = new Student();
                Student jian=new Student();
                bob.name = “bob”;
                bob.age=19;
                jian = bob;
                jian.name=”jian”;
```

```
                System.out.println(“bobs name: ”+bob.name);
        }
}
```

a) nothing prints
b) bobs name
c) bobs name bob
d) bobs name jian

respuesta, d, jian apunta a bob y se le cambio el nombre a esta mediante jian

69.- Which class has a default constructor?

```
class X{}
class Y{
        Y(){}
}
class Z{
        Z(int i){}
}
```

a) z only
b) x only
c) x, y and z
d) x and y
e) x and z
f) y only
g) y and z

respuesta, d, la clase x la tiene de forma implicita y y la tiene de forma explicita

70.- which two may precede the word “class” in a class declaration?

a) public
b) static
c) synchronized
d) local
e) volatile

71.- the builder pattern is used to:

a) having several constructor methods in a class
b) simplify the creation of complex objects
c) implement the constructor of a class

respuesta, b, el patron builder solo facilita la creacion de clases

72.- What is the result?

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

a) 0 1
b) 1 2
c) 2 2
d) 1 1

respuesta, b, esto porque primero se usa el dato antes de aumentarlo

73.- What value should replace kk in the comment to cause jj=5 to be output?

```
public class MyFive{
        public static void main(String[] args){
                short kk=11;
                short ii;
                short jj=0;
                for(ii=kk; ii>6; ii.=1){
                        jj++;
                }
                System.out.println("jj="+jj);
        }
}
```

a) -1
b) 1
c) 5
d) 8
e) 11

resultado, e, esto porque se anda restando uno al valor inicial y se espera que se ejecute al menos 5 veces, siendo 11 el unico valor que satisface

74.- What is a static block of code in java?

a) a block of code within a class that runs whenever the class is load on the jvm
b) a block of code inside a class that runs when that class first loaded in jvm
c) a block of code within a class that always runs before the builder

respuesta, b

75.-What is the result?

```
class X{
        X(){
                System.out.print(1);
        }
        X(int x){
                this();
                System.out.print(2);
        }
}
public class Y extends X{
        y(){
                super (6);
                System.out.print(3);
        }
        y(int y){
```

```
            this();
            System.out.print(4);
        }
        public static void main(String[] a){
            new Y(5);
        }
}
```

a) 2134
b) 4321
c) 2143
d) 13
e) 1213
f) 134

resultado, ninguna, es 1234, esto porque el programa va al constructor con entero de y, de ahi al constructor sin valores de y, de ahi al constructor con valores de x y termina llendo al constructor sin valores de x, de ahi imprime uno, sale del constructor sin valores de x y continua con el de valores de x que imprime dos, sale del constructor con valores de x y continua con el constructor sin valores de e imprime 3 para acabar con el constructor con valores de y e imprimir 4

76.- Which two actions, when used independently, will permit this class to compile?

```
import java.io.IOException;
public class Y{
        public static void main(String[] args){
            try{
                doSomething();
            }catch(RuntimeException e){
                System.out.println(e);
            }
        }
        static void doSomething(){
            if(Math.random()>0.5){
                throw new IOException();
            }
            throw new RuntimeException();
        }
}
```

a) adding throws IOException to the main() method …IOException
b) adding throws IOEXception to the doSomething() method signature and changing the catch
c) adding throws IOException to the main() method signature and to the doSomething() method
d) adding throws IOException to the main() method signature
e) adding throws IOException to the doSomething() method signature

resultados, b y c, es necesario añadirle el throws al método doSomething para IOException, pero tambien al añadirlo a la firma del método main se da una forma de lidiar con la excepcion, al igual que es el capturarla

77.- a method is declared to take three arguments. A program calls this method and passes only two arguments, what is the result?

a) compilation fails
b) the third argument is given the value void
c) the third argument is given the value zero
d) an exception occurs when the method attempts to access the third argument. the third argument is given the appropriate false value for its declared type. the third argument is given the value null.

resultado, a, la compilacion falla ya que se requiere que todos los parametros del método se encuentren presentes

78.- what is the result?

```
public class ScopeTest{
        int z;
        public static void main(String[] args){
                ScopeTest myScope=new ScopeTest();
                int z=6;
                System.out.print(z);
                myScope.doStuff();
                System.out.print(z);
                System.out.print(myScope.z);
        }
        void doStuff(){
                int z=5;
                doStuff2();
                System.out.print(z);
        }
        void doStuff2(){
                z=4;
        }
}
```

a) 6565
b) 6504
c) 6560
d) 6550

resultado, ninguna, es 6564 lo que imprime, esto debido a que la primera z que imprime es la variable z de main, al entrar a dostuff, vuelve a z interna en 4 para luego imprimir su propia variable interna z que es 5 para luego volver a main, donde imprime de nuevo la variable interna de main de z que sigue siendo 6 y despues la interna de la clase ScopeTest que es 4 debido a que cambio

preguntas del documento java_cuestionario2.pdf

1.- polimorfismo y excepciones
Considera el siguiente bloque de código

```
class Animal {
        void makeSound() throws Exception {
```

```
        System.out.println("Animal makes a sound");
        }
}
class Dog extends Animal {
        void makeSound() throws RuntimeException {
                System.out.println("Dog barks");
        }
}
public class Main {
        public static void main(String[] args) {
                Animal myDog = new Dog();
                try {
                        myDog.makeSound();
                } catch (Exception e) {
                System.out.println("Exception caught");
                }
        }
}
```

cual seria la salida en consola al ejecutar este código?

a) dog barks
b) animal makes a sound
c) exception caught
d) compilation error

resultado, a, toda excepcion esta bien manejada y el código compila cómo debe, en cuanto al comportamiento, myDog es un dog dentro de un animal, asi que al llamar al método en conjunto sobrecargado toma prioridad el de la clase que es, en este caso dog

2.- Hilos (threads)

considera el siguiente bloque de código:

```
class MyThread extends Thread {
        public void run() {
                System.out.println("Thread is running");
        }
}
public class Main {
        public static void main(String[] args) {
                Thread t1 = new MyThread();
                Thread t2 = new MyThread();
                t1.start();
                t2.start();
        }
}
```

a) thread is running (impreso una vez)
b) thread is running (impreso dos veces)
c) thread is running (impreso dos veces en orden aleatorio)
d) compilation error

respuesta, c, los threads andan bien declarados, asi que al momento de mandar llamarlos hacen el método declarado de forma aleatoria por naturaleza de hilos

3.- listas y excepciones
considera el siguiente bloque de código:

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

public class Main{
	public static void main(String[] args){
		List<Integer> numbers = new ArrayList<>();
		numbers.add(1);
		numbers.add(2);
		numbers.add(3);
		try{
			for (int i=0; i<=numbers.size();i++){
				System.out.println(numbers.get(i));
			}
		}catch(IndexOutOfBoundsException e){
			System.out.println(“Exception caught”);
		}
	}
}
```

cual seria la salida en consola al ejecutar este código?

a) 1 2 3 exception caught
b) 1 2 3
c) exception caught
d) 1 2 3 4

respuesta, a, al momento de llamar al indice 3 se genera un index out of bounds, el cual es atrapado

4.- Herencia, clases abstractas e interfaces
Considera el siguiente bloque de código:

```
interface Movable {
	void move();
}
abstract class Vehicle {
	abstract void fuel();
}
class Car extends Vehicle implements Movable {
	void fuel() {
		System.out.println("Car is refueled");
	}
	public void move() {
		System.out.println("Car is moving");
	}
}
public class Main {
```

```
	public static void main(String[] args) {
		Vehicle myCar = new Car();
		myCar.fuel();
		((Movable) myCar).move();
	}
}
```

a) car is refueled car is moving
b) car is refueled
c) compilation error
d) runtime exception

respuesta, a, al momento de declarar que myCar es un vehiculo pero construirlo cómo carro se tiene que usa los métodos definidos cómo su propia clase, en cuanto al move toma precedencia el definido ya que no tiene nada implementado la interfaz

5.- polimorfismo y sobrecarga de métodos
Considera el siguiente bloque de código:

```
class Parent {
	void display(int num) {
		System.out.println("Parent: " + num);
	}
	void display(String msg) {
		System.out.println("Parent: " + msg);
	}
}
class Child extends Parent {
	void display(int num) {
		System.out.println("Child: " + num);
	}
}
public class Main {
	public static void main(String[] args) {
		Parent obj = new Child();
		obj.display(5);
		obj.display("Hello");
	}
}
```

cual seria la salida en consola al ejecutar este código?

a) child 5 parent hello
b) parent 5 parent hello
c) child 5 child hello
d) compilation error

respuesta, a, al declararse cómo parent pero usar el constructor de child el objeto seria child, usando sus métodos propios pero si no tiene sus propios usa los de su padre, que en este caso seria el método que tiene una string cómo parámetro

6.- hilos y sincronizacion
considera el siguiente bloque de código:

```
class Counter {
	private int count = 0;
	public synchronized void increment() {
		count++;
	}
	public int getCount() {
		return count;
	}
}
class MyThread extends Thread {
	private Counter counter;
	public MyThread(Counter counter) {
		this.counter = counter;
	}
	public void run() {
		for (int i = 0; i < 1000; i++) {
			counter.increment();
		}
	}
}
public class Main {
	public static void main(String[] args) throws InterruptedException {
		Counter counter = new Counter();
		Thread t1 = new MyThread(counter);
		Thread t2 = new MyThread(counter);
		t1.start();
		t2.start();
		t1.join();
		t2.join();
		System.out.println(counter.getCount());
	}
}
```

cual seria la salida en consola al ejecutar este código?

a) 2000
b) 1000
c) variable count is not synchronized
d) compilation error

respuesta, a, al usar synchronized en increment se asegura que solo un hilo a la vez incremente el contador, evitando cualquier tipo de problema a la hora de ejecutar el programa, tambien al usar los joins se asegura que se terminen de ejecutar los hilos para imprimir el valor total del contador

7.- listas y polimorfismo

```
import java.util.ArrayList;
import java.util.List;
class Animal {
	void makeSound() {
		System.out.println("Animal sound");
```

```
        }
}
class Dog extends Animal {
        void makeSound() {
                System.out.println("Bark");
        }
}
class Cat extends Animal {
        void makeSound() {
                System.out.println("Meow");
        }
}
public class Main {
        public static void main(String[] args) {
                List<Animal> animals = new ArrayList<>();
                animals.add(new Dog());
                animals.add(new Cat());
                animals.add(new Animal());
                for (Animal animal : animals) {
                        animal.makeSound();
                }
        }
}
```

a) animal sound animal sound animal sound
b) bark meow animal sound
c) animal sound meow bark
d) compilation error

respuesta, b, se muestran cómo entraron, y al ser hijos dentro de padres se pueden meter dentro de la lista pero se mandan llamar los métodos propios de cada uno

8.- Manejo de excepciones y herencia
considera el siguiente bloque de código:

```
class Base {
        void show() throws IOException {
                System.out.println("Base show");
        }
}
class Derived extends Base {
        void show() throws FileNotFoundException {
                System.out.println("Derived show");
        }
}
public class Main {
        public static void main(String[] args) {
                Base obj = new Derived();
                try {
                        obj.show();
                } catch (IOException e) {
```

```
                System.out.println("Exception caught");
            }
        }
}
```

a) base show
b) derived show
c) exception caught
d) compilation error

respuesta, b, las excepciones se manejan bien, ya que se usa una del mismo package que IOException en el override de la clase hija, al momento de mandar llamar al método se usa el de la clase hija ya que sigue siendo una derived y no un objeto base

9.- concurrencia y sincronizacion
considera el siguiente bloque de código:

```
class SharedResource {
        private int count = 0;
        public synchronized void increment() {
                count++;
        }
        public synchronized void decrement() {
                count--;
        }
        public int getCount() {
                return count;
        }
}
class IncrementThread extends Thread {
        private SharedResource resource;
        public IncrementThread(SharedResource resource) {
                this.resource = resource;
        }
        public void run() {
                for (int i = 0; i < 1000; i++) {
                        resource.increment();
                }
        }
}
class DecrementThread extends Thread {
        private SharedResource resource;
        public DecrementThread(SharedResource resource) {
                this.resource = resource;
        }
        public void run() {
                for (int i = 0; i < 1000; i++) {
                        resource.decrement();
                }
        }
}
```

```
public class Main {
	public static void main(String[] args) throws InterruptedException {
		SharedResource resource = new SharedResource();
		Thread t1 = new IncrementThread(resource);
		Thread t2 = new DecrementThread(resource);
		t1.start();
		t2.start();
		t1.join(); t2.join();
		System.out.println(resource.getCount());
	}
}
```

cual seria la salida en consola al ejecutar este código?

a) 1000
b) 0
c) -1000
d) compilation error

respuesta b, ya que aumentan y decrementan al mismo tiempo y solo se aseguran de que terminen para mostrar el resultado, el resultado seria 0

10.- Generics y excepciones

Considera el siguiente bloque de código:

```
class Box<T> {
	private T item;
	public void setItem(T item) {
		this.item = item;
	}
	public T getItem() throws ClassCastException {
		if (item instanceof String) {
			return (T) item; // Unsafe cast
		}
		throw new ClassCastException("Item is not a String");
	}
}
public class Main {
	public static void main(String[] args) {
		Box<String> stringBox = new Box<>();
		stringBox.setItem("Hello");
		try {
			String item = stringBox.getItem();
			System.out.println(item);
		} catch (ClassCastException e) {
			System.out.println("Exception caught");
		}
	}
}
```

Cual seria la salida en consola al ejecutar este código?

a) hello
b) exception caught

c) compilation error
d) ClassCastException

respuesta, a, esto debido a que tanto los métodos dados cómo el manejo de la clase misma no causa ningun tipo de excepcion en ejecución, en cuanto a compilacion igual no hay ninguna

11.- ...

```
public class Main {
	public static void main(String[] args) {
		Padre objetoPadre = new Padre();
		Hija objetoHija = new Hija();
		Padre objetoHija2 = (Padre) new Hija();
		objetoPadre.llamarClase();
		objetoHija.llamarClase();
		objetoHija2.llamarClase();
		Hija objetoHija3 = (Hija) new Padre();
		objetoHija3.llamarClase();
	}
}
public class Hija extends Padre {
	public Hija() {
		// Constructor de la clase Hija
	}
	@Override
	public void llamarClase() {
		System.out.println("Llame a la clase Hija");
	}
}
public class Padre {
	public Padre() {
		// Constructor de la clase Padre
	}
	public void llamarClase() {
		System.out.println("Llame a la clase Padre");
	}
}
```

a) llame a la clase padre
   llame a la clase hija
   llame a la clase hija
   error: java.lang.classCastException
b) llame a la clase padre
   llame a la clase hija
   llame a la clase hija
c) llama a la clase padre
   llame a la clase hija
   llame a la clase hija
   llame a la clase padre

respuesta, a, primera y segunda son normales, la tercera castea solo al construir el objeto, no castea al momento de usar el método y la cuarta causa error porque no se puede castear a un hijo cómo padre

12.- …

```
import java.text.NumberFormat;
import java.text.ParseException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;
public class Ejemplos {
	public static void main(String[] args) {
		Animal uno=new Animal();
		Animal dos=new Dog();
		uno.makeSound();
		dos.makeSound();
		Dog tres=(Dog)new Animal();
		tres.makeSound();
	}
}
class Animal {
	void makeSound() {
		System.out.println("Animal sound");
	}
}
class Dog extends Animal {
	void makeSound() {
		System.out.println("Wau Wau");
	}
}
```

a) animal sound wau wau compilation error
b) compilation error
c) animal sound wau wau animal sound
d) animal sound

resultado, a, cómo no tal no causa un compilation error sino un class cast exception, pero es el mas cercano que hay

13.- …

```
import java.text.NumberFormat;
import java.text.ParseException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;
import java.lang.*;
public class Ejemplos {
	public static void main(String[] args) {
		Cambios uno=new Cambios();
		int x=1;
```

```
                String hola="hola";
                StringBuilder hola2=new StringBuilder("hola2");
                Integer x2=4;
                uno.makeSound(x, hola);
                uno.makeSound(x2, hola2);
                System.out.println("Cambios?: "+x+","+hola+","+x2+","+hola2);
        }
}
class Cambios{
        void makeSound(int x, String s) {
                s="cambiando string";
                x=5;
        }
        void makeSound(Integer x,StringBuilder s) {
                x=9;
                s=s.delete(0,s.length());
        }
}
```

a) compilation error
b) cambios? 1 hola 4
c) cambios? 1 hola 4 hola 2
d) cambios? 5 cambiando string 9

resultado, b, esto porque string no es mutable, si no se devuelve el resultado modificado se pierde y se mantiene la string sin modificacion, a diferencia de stringBuilder

14.- …

```
interface i1{
        public void m1();
}
interface i2 extends i1 {
        public void m2();
}
class animal implements i1,i2 {
        //¿Qué métodos debería implementar la clase animal en este espacio?
}
```

a) solo m1
b) m1 y m2
c) ninguno
d) error compilacion

resultado, b, m1 y m2 se necesitan declarar en animal porque se implementa tanto la interface 1 y 2

15.- …

```
class Animal {
        void makeSound() throws Exception {
                System.out.println("Animal makes a sound");
        }
}
```

```
class Dog extends Animal {
	void makeSound() throws RuntimeException {
		System.out.println("Dog barks");
	}
}
public class Main {
	public static void main(String[] args) {
		Animal myDog = new Dog();
		try {
			myDog.makeSound();
		} catch (Exception e) {
			System.out.println("Exception caught");
		}
	}
}
```

Cual seria la salida en consola al ejecutar este código?

a) dog barks
b) animal makes a sound
c) exception caught
d) compilation error

resultado, a, esto porque aunque este dentro de una clase animal, el objeto es dog y al llamar al método se comporta cómo dog

16.- …

```
import java.util.*;
import java.lang.*;
import java.io.*;
class Main {
	public static void main(String[] args) {
		String str = "1a2b3c4d5e6f";
		String []splitStr = str.split("//D");
		for(String elemento : splitStr){
			System.out.println(elemento);
		}
	}
}
```

imprime la cadena sin ningun tipo de alteracion, ya que la regex dada no la altera en nada

Preguntas Java_Cuestionario3.pdf

1.-Which declaration initializes a boolean variable?

a) boolean m=null
b) Boolean j=(1<5)
c) boolean k=0
d) boolean h=1

respuesta, b, es la unica que tiene una expresion booleana, ya que esta se evalua al momento de declarar

2.- What is the DTO pattern used for?

a) to exchange data between processes
b) to implement the data access layer
c) to implement the presentation layer

respuesta, a, dto es data transfer object, es un patron de diseño para transferir data entre objetos y asi reducir el numero de llamadas a sistema

3.-What value should replace kk in line 18 to cause jj=5 to be output?

```
public class MyFive{
        public static void main (String[] args){
                //short kk=7
                short ii;
                short jj=0;
                for(ii=kk; ii>6; ii-=1){
                        jj++;
                }
                System.out.println("jj="+jj);
        }
}
```

a) -1
b) 1
c) 5
d) 8
e) 11

respuesta, e, es la unica forma en la que se queda con 6, ya que reduce el valor total del original

4.- Cual sera el resultado?

```
public class SampleCalss{
        public static void main(String[] args){
                SampleClass sc, scA, scB;
                sc= new SampleClass();
                scA= new SampleClassA();
                scB= new SampleClassB();
                System.out.println("hash is: " + sc.getHash()+", "+scA.getHash()+",
"+scB.getHash();
        }
        public int getHash(){
                return 111111;
        }
}
class SampleClassA extends SampleClass{
        public int getHash(){
                return 999999999;
        }
}
```

a) compilation fails
b) an exception is thrown at runtime
c) there is no result because this is not the correct way to determine the hash code

d) hash is 111111, 44444444, 999999999

respuesta, d, aunque imprime esto y compila bien, no es una buena practica, el hash code debe ser unico para cada objeto

5.- cual seria el resultado?

```
public classDoCompare4 {
        public static void main(String[] args) {
                String[] table = {“aa”, “bb”, “cc”};
                int ii = 0;
                do {
                        while (ii < table.length) {
                                System.out.println(ii++);
                        }
                } while (ii < table.length);
        }
}
```

a) 0
b) 0 1 2
c) 0 1 2 0 1 2 0 1 2
d) compilation fails

respuesta, b, ya que ambos ciclos checan por la misma condicion para salir de ellos, cuando uno lo cumple el otro solo evalua antes de iniciar y sale

6.- cual seria el resultado?

```
public class DoCompare1{
        public static void main(String[] args){
                String[] table={“aa”,”bb”,”cc”};
                for (String ss: table){
                        int ii=0;
                        while (ii<table.length){
                                System.out.println(ss+”, “+ ii);
                                i++;
                        }
                }
        }
}
```

a) zero
b) once
c) twice
d) thrice
e) compilation fails

respuesta, d, imprime 3 veces todo debido a que son diferentes condiciones de salida, cuando un bucle acaba entra al siguiente

7.- What code should be inserted?

```
public class Bark{
        //insert code here line 5
                public abstract void bark();
```

```
        }
        //insert code here line 9
                public void bark(){
                        System.out.println("woof");
                }
        }
}
```

a) 5. class Dog { 9. public class Poodle extends Dog {
b) 5. abstract Dog { 9. public class Poodle extends Dog {
c) 5. abstract class Dog { 9. public class Poodle extends Dog {
d) 5. abstract Dog { 9. public class Poodle implements Dog { e)
e) 5. abstract Dog { 9. public class Poodle implements Dog {
f) 5. abstract class Dog { 9. public class Poodle implements Dog {

respuesta, c, la clase dog debe ser declarada cómo abstracta para que funcione la declaracion de la clase abstracta, la 9 es simplemente una declaracion de que poodle va a heredar de esta clase abstracta

8.- Which statement initializes a stringBuilder to a capacity of 128?

a) StringBuilder sb=new String("128");
b) StringBuilder sb=StringBuilder.setCapacity(128);
c) StringBuilder sb=StringBuilder.getInstance(128);
d) StringBuilder sb=new String(128);

9.-What is the result?

```
public class Calculator{
        int num=100;
        public void calc(int num){
                this.num=num*10;
        }
        public void printNum(){
                System.out.println(num);
        }
        public static void main(String[] args){
                Calculator obj=new Calculator();
                obj.calc(2);
                obj.printNum();
        }
}
```

a) 20
b) 100
c) 1000
d) 2

respuesta, a, el método recibe un valor que multiplica y mete en el num de la clase, el cual es despues llamado por printNum

10.- What three modifications, made independently, made to class Greet, enable the code to compile and run?

```
package handy.dandy;
```

```
public class KeyStroke{
        public class KeyStroke{
                public void typeExclamation(){
                        System.out.println(“!”);
                }
        }
}
```

And:

```
package handy;
public class Greet{
        public static void main(String[] args){
                String greeting = “hello”;
                System.out.print(greeting);
                KeyStroke.stroke= new KeyStroke();
                stroke.typeExclamation();
        }
}
```

a) Line 8 replaced with handy.dandy.KeyStroke stroke = new KeyStroke();
b) Line 8 replaced with handy.*.KeyStroke stroke = new KeyStroke();
c) Line 8 replaced with handy.dandy.KeyStroke stroke = new handy.dandy.KeyStroke();
d) import handy.*; added before line 1.
e) import handy.dandy.*; added after line 1.
f) import handy.dandy.KeyStroke; added after line 1.
g) import handy.dandy.KeyStroke.typeExclamation(); added after line 1.

respuesta, c, e y f, c ocupa toda la ruta explicita para la declaracion de la clase y asi poder crear una instancia correctamente, e importa todo lo que esta en handy.dandy y f importa especificamente la clase que se quiere usar

1.- consider the following java code snippet:

```
public int divide(int a, int b){
        int c= -1;
        try{
                c=a/b;
        }catch(Exception e){
                System.err.print(“Exception “);
        }
        finally{
                System.err.println(“finally “);
        }
        return c;
}
```

what will our code print when we call divide 4,0?

a) exception finally
b) finally exception
c) exception

respuesta, a, se muestra una exception por dividir entre cero y luego se imprime finally por que es lo normal

2.-The feature which allows different methods to have the same name and arguments type, but different implementation is called?

a) overloading (sobre carga)
b) overriding (sobre escritura @override)
c) java does not permit methods with same and type signature
d) none of the above

respuesta, b

3.- what does the following for loop output?

```
for(int i=10, j=1;i>j;--i,++j)
        System.out.print(j %i);
```

a) 12321
b) 12345
c) 11111
d) 0000

respuesta, b, al ser el menor mod mayor, imprimiria los valores que tiene j

4.- we perform the following sequence of actions:
1.- insert the following elements into a set 1,2,9,1,2,3,1,4,1,5,7
2.- convert the set into a list and sort it in ascending order.
which options denotes the sorted list?

a) {1,2,3,4,5,7,9}
b) {9,7,5,4,3,2,1}
c) {1,1,1,1,2,2,3,4,5,7,9}
d) none of the above

respuesta, a, el set no repite miembros

5.- what is the output for the below java code?

```
public class Test{
        public static void main(String[] args){
                int i=010;
                int j=07;
                System.out.println(i);
                System.out.println(j);
        }
}
```

a) 8 7
b) 10 7
c) compilation fails with an error at line 3
d) compilation fails with an error at line 5

6.- A public data member with the same name is provided in both base as well as derived clases. which of the following is true?

a) it is a compiler error to provide a field with the same name in both base and derived class
b) the program will compile and this feature is called overloading
c) the program will compile and this feature is called overriding
d) the program will compile and this feature is called as hiding or shadowing

respuesta, d

7.- which statement is true?

a) non static member classes must have either default or public accessibility
b) all nested classes can declare static member classes
c) methods in all nested classes can be declared static
d) static member classes can contain non-static methods

respuesta, d

8.- a constructor is called whenever

a) an object is declared
b) an object is used
c) a class is declared
d) a class is used

respuesta, a, es llamado cuando el objeto es declarado, mas especificamente cuando se crea la instancia de este, no es cuando se usa

9.- Which of the following data types in java are primitive?

a) String
b) Struct
c) boolean
d) Char

respuesta, d, string y struct son compuestos mientras que boolean no es primitivo

10.- which of the following are true for java classes?

a) the void class extends the class class
b) the float class extends the double class
c) the system class extends the runtime class
d) the integer class extends the number class

respuesta, d, todos los wrappers de numeros extienden de number

11.- the following code snippet is a demonstration of a particular design pattern, which design pattern is it?

```
public class Mystery{
        private static Mystery instance=null;
        protected Mystery()[
                public static Mystery getInstance(){
                        if(instance==null){
                                instance = new Mystery();
                        }
                        return instance;
                }
        }
}
```

a) factory design pattern
b) strategy pattern
c) singleton
d) facade design pattern

respuesta, c, se emplea el modelo singleton porque se maneja una unica instancia a lo largo de la ejecución

12.- which of the following java declaration of the String array is correct?

a) String temp[] = new String{“j”,”a”,”z”};
b) String temp[]={“j” “b” “c”}
c) String temp={“a”, “b”, “c”};
d) String temp[] = {“a”, “b”, “c”};

respuesta, d, esto porque se tiene tanto la declaracion de que es array y la separacion por comas de cada elemento

13.-which is true of the following program?

```
package exam.java;
public class TestFirstApp{
        static void doIt(int x, int y, int m){
                if(x==5) m=y;
                else m=x;
        }
        public static void main(String[] args){
                int i=6,j=4,k=9;
                TestFirstApp.doIt(i,j,k);
                System.out.println(k);
        }
}
```

a) doesnt matter what the values of i and j are, the output will always be 5
b) doesnt matter what the values of k and j are, the output will always be 5
c) doesnt matter what the values of i and j are, the output will always be 9
d) doesnt matter what the values of k and j are, the output will always be 9

resultado, d, esto porque no sobreescribe el valor de las variables en ninguna forma

14.- Which of the following statements are correct. Select the correct answer.

a) Each Java file must have exactly one package statement to specify where the class is stored.
b) If a java file has both import and package statement, the import statement must come before package statement.
c) A java file has at least one class defined
d) If a java file has a package statement, it must be the first statement (except comments).

resultado, d, no es a porque cualquier clase puede tener multiples packages, no es b porque primero debe ir el package, no es c porque puede tener solo enums o algo por el estilo

15.- Given the following code, what is the most likely result.

```
import java.util.*;
public class Compares{
        public static void main(String[] args){
                String[] cities={“bangalore”, “pune”, “san francisco”, “new york city”};
                MySort ms= new MySort();
                Arrays.sort(cities.ms);
                System.out.println(Arrays.binarySearch(cities,”new york city”));
        }
        static class MySort implements comparator{
                public int compare(String a,String b){
```

```
                    return b.compareTo(a);
                }
        }
}
```

a) 1
b) 1
c) 2
d) compilation fails

resultado, d, falla la compilacion porque se usa el erroneo, se debe usar comparable si se quiere comparar de esta forma, comparator es para comparar dos objetos dados

16.- To delete all pairs of keys and values in a given HashMap, which of the following methods should be used?

a) clearAll()
b) empty()
c) remove()
d) clear()

respuesta, d, es el método correcto para vaciar un hash map

17.- Which pattern do you see in the code below:

java.util.Calendar.getInstance();

a) Singleton Pattern
b) Factory Pattern
c) Facade Pattern
d) Adaptor Pattern

resultado, b, porque tiene el método de getInstance

18.- What is the output of the following program:

```
interface Base{void method();}
class BaseC{
        public void method(){
                System.out.println(“inside baseC method”);
        }
}
class ImplC extends BaseC implements Base{
        public static void main(String[] s){
                new implc().method();
        }
}
```

a) null
b) compilation fails
c) inside baseC method
d) none of the above

resultado, c, toma el método extendido primero

19.- Consider the following three classes:

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

consider n object of class B is instantiated, i.e.
B b = new B();
which of the following boolean expressions evaluates to true?

a) (b instanceof B)
b) (b instanceof B) && (!(b instanceof A))
c) (b instanceof B) && (!(b instanceof C))
d) none of the above

resultado, a y c, con a es true debido a que es auto explicatorio, con c es true debido a que no es instancia de c, volviendola true con la negacion

20.- What is the output of the following program:

```
class Constructor{
        static String str;
        public void Constructor(){
                System.out.println(“in a constructor”);
                str=”hello world”;
        }
        public static void main(String[] args){
                Constructor C= new Constructor();
                System.out.println(str);
        }
}
```

a) in constructor
b) null
c) compilation fails
d) none of the above

resultado, null, esto porque no cumple con los requisitos para ser constructor, que es que no devuelva nada, en este caso indica que devuelve void, que aunque no es nada es algo para la declaracion de constructor

Preguntas de paola

1.- Given:

```
public class MyFor3{
        public static void main(String[] args){
                int[] xx= null;
                System.out.println(xx);
        }
}
```

what is the result?

a) null
b) compilation fails
c) java.lang.NullPointerException
d) 0

respuesta, a, no causa ningun error ni complicacion al compilar ponerle null, a menos que se intente usar un método interno no hay problema

2.-given a java source file:

```
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();
        }
}
```

what changes will make this code compile?

a) adding the public modifier to the declaration of class X
b) adding the protected modifier to the x() constructor
c) changing the private modifier on the declaration of the one() method to protected
d) removing the Y() constructor

resultado, c, no compila porque no encuentra el método one porque este es privado para x, al cambiarlo a protected compila sin ningun problema

8.- Given the code fragment:

```
String h1= “bob”
String h2=new String(“bob”);
```

what is the best way to test that the values of h1 and h2 are the same?

a) if (h1==h2)
b) if(h1.equals(h2))
c) if(h1==h2)
d) if(h1.same(h2))

resultado, b, al no estar en pool de strings, la mejor forma de saber si son iguales es con equals

9.- Given the code fragment:

```
String color=”red”;
switch(color){
        case “red”:
                System.out.println(“found red”);
        case “blue”:
                System.out.println(“found blue”);
                break;
        case “white”:
                System.out.println(“found white”);
                break;
        default:
                System.out.println(“found default”);
}
```

what is the result?

a) found red

b) found red found blue
c) found red found blue found white
d) found red found blue found white found default

resultado, b, debido a que red no tiene break se sigue hasta blue que tiene break para salir del switch

10.- Given:

```
public class Bark{
        //insert code here line 5
                public abstract void bark();// line 6
        }//line 7
        //line 8
        //insert code here line 9
public void bark(){
        System.out.println("woof");
}
}
```

a) 5 class Dog{ 9 public class Poodle extends Dog{
b) 5 abstract Dog{ 9 public class poodle extends Dog{
c) 5 abstract class Dog{ 9 public class Poodle extends Dog{
d) abstract Dog{ 9 public class Poodle implements Dog

respuesta, c, tiene que ser una clase abstracta si usa clases abstractas, y a diferencia de b, la c lo implementa bien

11.- Given the code fragment:

```
int j=0,k=0;
for(int i=0;i<x;i++){
        do{
                k=0;
                while(k<z){
                        k++;
                        System.out.println(k+"");
                }
                System.out.println("");
                j++;
        }while(j<y);
        System.out.println("---");
}
```

what values of x,y,z will produce the following result?

```
1 2 3 4
1 2 3 4
1 2 3 4
---------
1 2 3 4
---------
```

a) x=2, y=3, z=4
b) x=3, y=2, z=3
c) x=2, y=3, z=3

d) x=4, y=2, z=3

resultado, a, debido a que los valores del 1 al 4 se imprimen, z debe ser 4, y al ser la unica opcion con z=4 entonces es esa

12.- Given:

```
class X{}
class Y{Y(){}}
class Z{z(int i){}}
```

which class has a default constructor?

a) x only
b) x and z
c) z only
d) x and y

resultado, a, ya que x no tiene constructor se manda llamar al default, y por otra parte tiene un constructor identico al default pero no es el default cómo tal

13.- Given:

```
class Overloading{
        int x(double d){
                System.out.println(“one”);
                return 0;
        }
        String x(double d){
                System.out.println(“two”);
                return null;
        }
        double x(double d){
                System.out.println(“three”);
                return 0.0;
        }
        public static void main(String[] args){
                new Overloading().x(4.0);
        }
}
```

what is the result?

a) one
b) two
c) three
d) compilation fails

resultado, d, falla la compilacion porque todos reciben el mismo tipo de parámetro

14.-Given:

```
public class X implements Z{
        public String toString(){
                return “x”;
        }
        public static void main(String[] args){
                Y myY=new Y();
```

```
                X myX= myY;
                Z myZ=myX;
                System.out.prin(myX);
                System.out,print((Y)myX);
                System.out.print(myZ);
        }
}
class Y extends X{
        public String toString(){
                return “Y”;
        }
}
interface Z{}
```

a) x x x
b) x y x
c) y y x
d) y y y

resultado, d, se copia el mismo objeto en los 3, siendo casteado en la segunda y en la ultima solo llamado, cómo z es abstracta, no puede implementar un toString y usa el del objeto

15.-given the code fragment:

```
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]);
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]);
```

a) 4 null
b) null 4
c) an illegalArgumentException is thrown at run time
d) 4 an arrayIndexOutOfBoundsException is thrown at runtime

respuesta, d, no hay un indice 4 en el array con indice 1

16.- given:

```
public class MyFor{
        public static void main(String[] args){
                for(int ii=0; ii<4; ii++){
                        System.out.println(“ii= “+ii);
                        ii=ii+1;
                }
        }
}
```

what is the result?

a) ii=0 ii=2
b) ii= 0 ii= 1 ii= 2 ii=3
c) ii=

d) compilation fails

resultado, a, debido a la suma extra que se hace despues de imprimir, solo imprime 0 y 2

17.- given

```
String message1=”wham bam”;
String message2= new String (“wham bam”);
if(message1==message2)
        System.out.println(“match”);
if(message1.equals(message2))
        System.out.println(“really match”);
```

what is the result?

a) match really match
b) really match
c) match
d) nothing prints

resultado, b, por no estar en pool de strings no se consideran mismos objetos, pero equals checa contenido

18.- Given:

```
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{}
```

a) asc=sc;
b) sc=asc;
c) asc=(object)sc;
d) asc=sc.clone();

respuesta, b, padre puede contener a hijos, en este caso sampleclass es padre de anothersampleclass

19.-given the code fragment:

```
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]);
```

a) 3false1
b) 2true3
c) 2false3
d) 3true1

respuesta, d, la longitud del indice 0 es 3, getClass().isArray() es true porque es array

20.-given the fragment

```
int[] array={1,2,3,4,5};
System.arraycopy(array,2,array,1,2);
```

System.out.print(array[1]);
System.out.print(array[4]);
what is the result?

a) 35
b) 15
c) 24
d) 25

resultado, a, el arraycopy deja el array cómo [1,3,4,4,5]

21.- given the code fragment
String name=”spot”;
int age=4;
String str=”my dog”+name+”is”+age;
System.out.prinln(str);
And:
StringBuilder sb=new StringBuilder();
using StringBuilder, which code fragment is the best potion to build and print the following String My dog spot is

a) sb.append(“my dog”+name+”is”+age); System.out.println(sb);
b) sb.insert(“my dog”).append(name+”is”+age); System.out.println(sb);
c) sb.insert(“my dog”).insert(name).insert(“is”).insert(age);
d) sb.append(“my dog”).append(name).append(“is”).append(age);
System.out.println(sb);

respuesta, a y d, esto porque stringbuilder no tiene insert

22.-given:

```
public class DoBreak1{
        public static void main(String[] args){
                String[] table){“aa”,”bb”,”cc”,”dd”};
                for(String ss:table){
                        if(“bb”.equals(ss)){
                                continue;
                        }
                        System.out.println(ss);
                        if(“cc”.equals(ss)){
                                break;
                        }
                }
        }
}
```

what is the result?

a) aa cc
b) aa bb cc
c) cc dd
d) cc

resultado, a, esto porque se salta la impresión de bb debido al primer if, luego evita la impresión de dd al entrar al segundo if.

23.- Which three are valid types for switch?

a) int
b) float
c) Integer
d) String

respuesta, a, c y d, el switch acepta los no punto flotante y char y string