archivo questions2daVuelta.pdf

1.- the builder pattern is used to: Simplify the creation of complex objects

Hace esto al separar la construccion del objeto de la instanciacion del mismo, haciendo una clase que se dedique a crear el objeto y otra que solo se dedique a instanciar el mismo con un objeto de la clase que se dedico a crear al objeto

2.- In Java the difference between throws and throw is:

throws indicates the type of exception that the method does not handle and throw an exception

throws manda la excepcion afuera del método para ser manejada, throw solo devuelve la excepcion

3.- A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the result? Compilation fails

Falla compilacion por la falta de variables necesarias.

4.-Which three implementations are valid?

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

A)
```
class Test implements SampleCloseable{
        public void close() throws java.io.IOException){
        //do something
        }}
```

B)
```
class Test implements SampleCloseable{
        public void close() throws Exception{
                //do something
                }}
```

C)
```
class Test implements SampleCloseable{
        public void close() throws FileNotFoundException{
                //do something
                }}
```

D)
```
class Test extends SampleCloseable{
        public void close() throws java.io.IOException{
                //do something
                }}
```

E)
```
class Test implements SampleCloseable{
        public void close(){
                //do something
                }}
```

a, c, e, esto debido a que al hacer overload de la interfaz pueden mandar llamar la misma excepcion (a), pueden llamar a una excepcion hija (FileNotFoundException, c) o pueden no llamar a ninguna (e)

5.- 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 Tell();
                speakIT.tellItLikeItIs();//no
                (Truth)speakIT.tellItLikeItIs();//no
                ((Truth)speakIT).tellItLikeItIs();//yes
```

```
                tellIt.tellItLikeItIs();//yes
                (Truth) tellIt.tellItLikeItIs();//no
                ((Truth)tellIt).tellItLikeItIs();//yes
        }
}
class Tell extends Speak implements Truth{
        @Override
        public void tellItLikeItIs(){
                System.out.println("Right On!");
        }
interface Truth{
        public void tellItLikeItIs();
}
```

las marcadas cómo no, no compilan, entre ellas esta que speak no tiene un método tellItLikeItIs y que se anda llamando a el sin ningun tipo de casteo previo, luego se tiene dos casteos no validos ya que andan casteando al resultado del método tellItLikeItIs y no al tellIt

6.- What changes will make this code compile? Select 2

```
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 declaration of the one() method to protected //si
Removing the Y() constructor

Al cambiar el modificador de privado en la declaracion del método de one a protected, permite que Y sepa que existe y pueda acceder a el.

7.- Que imprime?

```
public class Main(){
        public static void main(String[] args){
                int x=2;
                for(;x<5;){
                        x=x+1;
                        System.out.println(x);
                }
        }
}
```

3
4
5

8.- Cual es el resultado?

```
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});
	}
}
```

Los prints tienen un error al ser llaves en vez de corchetes, si se ignora esto, causa un out of bounds despues de imprimir un 4.

Which two possible outputs?

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

primer caso, random>0.5 imprimiria before if clause para luego mostrar la exception lanzada
segundo caso, random<0.5 imprimiria before if clause y after if clause

9.- Cual es la salida?

```
public class Main{
	public static void main(String[] args){
		int x=2;
		if(x==2) System.out.println(“A”);
		else System.out.println(“B”);
		else System.out.println(“C”);
	}
}
```

compilation error por código inalcanzable despues del if, ya que no hay forma alguna de alterar el resultado de x

10.- How many times is 2 printed?

```
class Menu{
	public static void main(String[] args){
		String[] breakfast={“beans”, “egg”, “ham”, “juice”};
		for(String rs : breakfast){
			int dish=1;
			while (dish<breakfast.length){
```

```
                                System.out.println(rs+","+dish);
                                ++dish;
                        }
                }
        }
}
```

si se ignora el que faltan los parentesis del breakfast.length(), seria beans, 1, beans, 2, beans 3, egg, 1, egg, 2, egg, 3, ham, 1, ham, 2, ham, 3, juice, 1, juice, 2, juice, 3
se imprime 2 4 veces en toda la ejecución.

11.- 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){//18
        }
}
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

La respuesta es d, no hace llamada al constructor del padre para crear la clase, debe hacerlo, esto es con super("id")

12.- 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

la respuesta es f, ya que manda llamar primero al constructor de la clase padre al momento de crear al hijo, que en este caso montaña manda llamar roca que manda llamar a atom, siendo atom primero impreso para luego imprimir el print de roca, lo mismo pasa en la siguiente linea

13.- 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

La respuesta es c, parse lo que hace es sólo convertir la string en un objeto número, no la cambia para nada con los valores que tiene para darle formato a los números, para hacer esto se ocuparia usar nf.format(Double.parseDouble(s))

14.-What is the result?

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

error de compilacion al no tener mph definido, tambien al no implementar getTicket
si se cambia mph para que sea string, y se cambia el nombre del método getTicket a getGait, el unico problema restante seria volver public al método getGait para que este en override

16.- Which two actions, 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 signature and to the doSomething() method.
B) Adding throws IOException to the main() method … IOException
C) Adding throws IOException to the doSomething() method signature
D) Adding throws IOException to the main() method signature
E) Adding throws IOException to the doSomething() method signature and changing the catch

ya que se requiere el throws en la firma del método de forma necesaria cuando se lanza una excepcion, solo a y e tienen sentido, la a es mas sobre pasarle la excepcion a alguien mas y no manejarla ella misma, mientras que la e habla de cambiar el catch para atrapar la IOException

17.-

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

a y d, ya que al hacer rollback al primer punto de guardado ya se tiene que esta en +10 la cuenta, mientras que al hacer rollback al segundo punto de guardado se tiene que ya fue +10 de la primera pero tambien +100 de la segunda

19.- Cuales de las siguientes opciones son validas?

A) el constructor predeterminado proporcionado por el compilador puede ser llamado utilizando this().
B) un constructor puede ser invocado desde un método de instancia.
C) una variable de instancia puede ser accedida dentro de un método de clase (static)
D) un constructor puede ser llamado dentro de otro constructor utilizando la palabra clave this().

Las respuestas son b y d, la b es el principio del singleton, ya que es un método el que genera la unica instancia que es llamada entre clases y en caso de que exista, regresa null Mientras que la d es una forma de llamar a otro tipo de constructores con valores predeterminados en caso de que llegase a faltar un dato, llenando el faltante con valores default ya sea de java o default para la definicion de la clase

20.- Cual es el resultado?

```
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

la respuesta correcta es b, ya que se altera el resultado de la variable pasada pero al ser solo una copia local del método, no hay forma alguna de que esta se guarde en memoria.

21.- what is the result if the integer value is 33?

```
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("...");

            }
        }
}
```

A) the fox jump lazy?
B) the fox lazy?
C) quick fox over lazy?

respuesta, the fox lazy, si se sigue la cadena de ifs, se pueden elegir hasta 4, por el valor dado de 33, al evaluar en el primer if da imprime the, al evaluar en segundo no pasa, al evaluar en el tercero se queda en la primera condicion de fox, al evaluar en el ultimo if imprime lazy

What is the result if you try to compile Truthy.java and then run it with assertions enabled?

```
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.java compiles and an AssertionError is thrown with x!=6 as additional output
C) Truthy.java does not compile
D) Truthy.java compiles and an AssertionError is thrown with no additional output

respuesta es c, asserts necesitan que se devuelva boolean

25.- Definicion de singleton
Patrón de diseño que permite tener una sola instancia de clase la cual es accedida por todos los que tienen acceso a la instancia

Diferencias entre clases abstractas e interfaces

Una clase puede heredar de una interfaz? no, son implementadas
si una clase puede heredar de multiples clases? no, solo una
si en una interfaz puede haber métodos con comportamiento y default? se puede

Preguntas de questions3eravuelta

70.- Selecciona la respuesta correcta con respecto al resultado del bloque de código.

```
public class Test5{
	public static void main(String argos[]){
		Side primerIntento=new Head();
		Tail segundoIntento = new Tail();
		Coin.overload(primerIntento);
		Coin.overload((Object)segundoIntento);
		Coin.overload(segundoIntento);
		Coin.overload((Side)primerIntento);
	}
}
interface Side{ String getSide();}
class Head implements Side{
	public String getSide(){
		return “Head”;
		}
	}
class Tail implements Side{
	public String getSide(){
		return “Tail”;
		}
	}
class Coin{
	public static void overload(Head side){
		System.out.println(side.getSide());
		}
	public static void overload(Tail side){
		System.out.println(side.getSide());
		}
	public static void overload(Side side){
		System.out.println(“side”);
		}
	public static void overload(Object side){
		System.out.println(“object”);
		}
	}
```

imprime side, object, tail, side, esto debido a que primer intento es tipo side, no head y asi entra al overload, en el segundo entra cómo object asi que imprime lo que esta en object, lo mismo con el ultimo ya que entra cómo side

18.-What is the result?
Given

```
public class SuperTest{
	public static void main(String[] argos){
		//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 the 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();

tiene que ser vacio, dejando c,d,e cómo opciones, pero el ultimo caso hace que sea d la respuesta correcta

21.- Cuales de las siguientes opciones son instanciaciones e inicializaciones validas de un arreglo multidimensional? elige dos

A) int[][] array2d={{0,1,2,4},{5,6}}
B)
```
int[][] array2D =new int[][2];
array2d[0][0]=1;
array2d[0][1]=2;
array2d[1][0]=3;
array2d[1][1]=4;
int[] array3d = new int[2][2][2];
```

C) int[][][] array3d={{{0,1},{2,3},{4,5}}};
int[] array={0,1}

D) array3d[0][0]=array;
array3d[0][1]=array;
array3d[1][0] =array;
array3d[1][1]= array;

a y c, b no se puede porque se debe declarar minimo el tamaño de la primera dimension, d no es porque no funciona asi la declaracion

??.-Que imprime?

```
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();
	}
}
```

imprime 20

??.- que imprime?

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

imprime feline, cougar, c c

??.- 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) 27 mph, lope
D) compilation fails

es d, no compila porque getGait debe estar cómo publico

45.- 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;

a, e y f son correctas, b no se puede porque son diferentes sin ningun tipo de relacion entre ellas mas que padre, c no se puede meter a un padre a un hijo, solo viceversa, d mismo caso que b aun casteandolo

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

es la d, el default puede ser considerado cómo lo pone la clase Y, la z no cuenta porque ya recibe un parámetro alejandolo de la default

??.- Given:
Map<String, List<? extends CharSequence>> stateCitiesMap = new HashMap<String,List<? extends CharSequence>>();
Which of the following options correctly achieves the same declaration using type inference

A) Map<String,List<? extends CharSequence>> stateCitiesMap=new HashMap>String,List>>();
B) Map<String,List<? extends CharSequence>> stateCitiesMap=new HashMap();
C) Map<String,List<? extends CharSequence>> stateCitiesMap=new HashMap<>();
D) Map<String,List<? extends CharSequence>> stateCitiesMap=new HashMap<<>>();

c?

??.- What will the following code print?

```
int i=1;
int j=i++;
if((i==++j)|(i++==j)){
        i+=j;
}
System.out.println(i);
```

imprime 5

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

a)
```
int max(int x, int y){
        return(if(x>y){ x;}else{y;});
}
```
b)
```
int max(int x, int y){
        return(if(x>y){ return x;}else{return y;});
}
```
c)
```
int max(int x, int y){
        switch(x<y){
                case true:
                        return y;
```

```
                        case false:
                                return x;
                };
        }
```

d)
```
int max(int x, int y){
        if(x>y) return x;
        return y;
}
```

la respuesta es d, a no regresa nada, b intenta regresar algo pero esta mal implementado por el primer return, y con c switch no permite booleanos

??.- What will the following code print when run without any arguments?

```
public class TestClass{
        public static int m1(int i){
                return ++i;
        }
        public static void main(String[] args){
                int k=m1(args.length);
                k+= 3 + ++k;
                System.out.println(k);
        }
}
```

A) it will throw ArrayIndexOutOfBoundsException
B) it will throw NullPointerException
C) 6
D) 5
E) 7
F) 2

la respuesta es c, args.length es 0, se le asigna a k cómo 1 por m1, la expresion k+= 3 + ++k es equivalente a k= 3 + 2 + 1

??.- What will be the result of attempting to compile and run the following code?

```
public class PromotionTest{
        public static void main(String args[]){
                int i=5;
                float f=5.5f;
                double d=3.8;
                char c='a';
                if(i==f) c++;
                if(((int)(f+d))==((int) f+ (int)d)) c+=2;
                System.out,println(c);
        }
}
```

imprime a, no entra al if debido a que al hacer la suma mientras son punto flotante da otro resultado que al castearlos individualmente mientras se suman.

??.- What letters will be printed by this program?

```
public class ForSwitch{
```

```
	public static void main(String[] args){
		char i;
		LOOP: for(i=0;i<5;i++){
			switch(i++){
				case ‘0’: System.out.println(“A”);
				case 1: System.out.println(“b”); break LOOP;
				case 2: System.out.println(“c”); break;
				case 3: System.out.println(“d”);break;
				case 4: System.out.println(“e”);
				case ‘E’: System.out.println(“f”);
			}
		}
	}
}
```

i entra cómo 0 al primer bucle, no hay caso 0 asi que no pasa nada en switch pero se le suma uno, volviendolo 1, para luego salir de ese bucle cómo 2, entra cómo 2 al switch imprimiendo C pero sumandole uno mas, volviendolo 3 al terminar el switch, que se vuelve 4 al terminar el bucle de for, entra cómo 4 una ultima vez al for, siendo comparado cómo 4 en el switch imprimiendo e y f por falta de break, sale del switch cómo 5, sumandole uno mas por el for volviendolo 6, saliendo del for porque es mayor que 5.
En resumen, imprime C E F

??.- Consider the following class:

```
public class Test{
	public static void main(String[] args){
		if(args[0].equals(“open”))
			if(args[1].equals(“someone”))
				System.out.println(“Hello!”);
		else System.out.println(“Go away”+args[1]);
	}
}
```

which of the following statements are true if the above program is run with the command line: java Test closed

A) it will throw ArrayIndexOutOfBoundsException at runtime
B) it will end without exceptions and will print nothing
C) it will print go away and then will throw ArrayIndexOutOfBoundsException
D) none of the above

respuesta b, al solo checar el primer parámetro de args, el cual es closed, solamente termina el programa ya que no entra al if, no da oportunidad de causar una excepcion debido a que solo si encuentra que sea open causa la excepcion

??.- How many objects have been created by the time the main method reaches its end in the following code?

```
public class Noobs{
	public Noobs(){
	try{
```

```
                throw new MyException();
        }catch(Exception e){
        }
}
public static void main(String[] args){
        Noobs a= new Noobs();
        Noobs b= new Noobs();
        Noobs c=a;
}
class MyException extends Exception{}}
```

2??

??.- What will the following code print?

```
public class TestClass{
        static char ch;
        static float f;
        static boolean bool;
        public static void main(String[] args){
                System.out.print(f);
                System.out.print(“ “);
                System.out.print(ch);
                System.out.print(“ “);
                System.out.print(bool);
        }
}
```

imprime 0.0, tres espacios y false, todo junto

??.- Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest{
        String a=”x”;
        static char b=’x’;
        String c=”x”;
        class Inner{
                String a=”y”;
                String get(){
                        String c=”temp”;
                        //line 1
                        return c;
                }
        }
        AccessTest(){
                System.out.println(new Inner().get() );
        }
        public static void main(String args[]){
                new AccessTest();
        }
```

}

a) c=c;
b) c=this.a;
c) c=""+AccessTest.b;
d) c=AccessTest.this.a;
e) c=""+b;

c d y e son las respuestas correctas, porque a solo reasignaria temp a si misma, ya que le falta el AccessTest.this para que se distinga de la variable interna, b es parecido pero con la variable interna del método

??.- What will be the result of attempting to compile and run the following program?

```
public class TestClass{
        public static void main(String args[]){
                        int x=0;
                        labelA: for(int i=10;i<0;i--){
                                int j=0;
                                labelB:
                                while(j<10){
                                        if(j>i) break labelB;
                                        if(i==j){
                                                x++;
                                                continue labelA;
                                        }
                                        j++;
                                }
                                x--;
                        }
                        System.out.println(x);
                }
}
```

imprime 0, no entra siquiera al for porque la condicion de entrada es que sea menor que 0 y esta siendo declarada cómo 10.

??.- cual de las siguientes opciones compila?

```
abstract class Vehicle{}
interface Drivable{}
class Car extends Vehicle implements Drivable{}
class SUV extends Car{}
```

A) ArrayList<Vehicle> all = new ArrayList<>();
   SUV s=a11.get(0);
B) ArrayList<Drivable> a12=new ArrayList<>();
   Car c1=a12.get(0);
C) ArrayList<SUV> a13 = new ArrayList<>();
   Drivable d1 = a13.get(0);
D) ArrayList<SUV> a14 = new ArrayList<>();
   Car c2=a14.get(0);

E) ArrayList<Vehicle> a15 = new ArrayList<>();
Drivable d2=a15.get(0);

las respuestas son c y d, esto mas por cuestion de que car y drivable son padres o implementan de alguna forma la interface

??.- Que imprime?

```
public class Speak{
	public static void main(String[] args){
		Speak speakIT=new Tell();
		Tell tellIT= new Tell();
		speakIT.tellItLikeItIs();
		(Truth)speaktIT.tellITLikeItIs();
		((Truth)speakIT).tellItLikeItIs();//correcta
		tellIt.tellItLikeItIs();
		(Truth) tellIt.tellItLikeItIs();
		((Truth)tellIT).tellItLikeItIs();//correcta
	}
}
class Tell extends Speal implements Truth{
	@Override
	public void tellItLikeItIs(){
		System.out.println(“Right on!”);
	}
}
interface Truth{
	public void tellItLikeItIs();
}
```

Las marcadas cómo correctas son las que si corren en el programa, las demas estan incorrectas

??.- que imprime?

```
public class Test5{
	public static void main(String args[]){
		Side primerIntento=new Head();
		Tail segundoIntento=new Tail();
		Coin.overload(primerIntento);
		Coin.overload((Object)segundoIntento);
		Coin.overload(segundoIntento);
		Coin.overload((Side)primerIntento);
	}
}
interface Side{
	String getSide();
	}
class Head implements Side{
	public String getSide(){
```

```
                return “head”;
        }
}
class Tail implements Side{
        public String getSide(){
                return “Tail”;
        }
}
class Coin{
        public static void overload(Head side){
                System.out.println(side.getSide());
                }
        public static void overload(Tail side){
                System.out.println(side.getSide());
                }
        public static void overload(Side side){
                System.out.println(“side”);
                }
        public static void overload(Object side){
                System.out.println(“object”);
                }
        }
```

Imprime side object tail side

Preguntas del documento javacuestionario4.pdf

1.-Que arroja?

```
public class Main{
        public static void main(String[] args){
                String[] at = {“FINN”,”JAKE”};
                for (int x=1;x<4;x++){
                        for (String s:at){
                                System.out.println(x+” ”+s);
                                if(x==1){
                                        break;
                                }
                        }
                }
        }
}
```

1 finn 2 finn 2 jake 3 finn 3 jake
imprime en la primera vuelta, 1 finn, esto debido a que sale del ciclo una vez que termina de imprimir debido al if(x==1), lo demas sigue el flujo normal del programa

2.- Que 5 lineas son correctas?

```
class Light{
        protected int lightsaber(int x){return 0;}
```

```
}
class Saber extends Light{
	private int lightsaber(int x){return 0;} //1
	protected int lightsaber(long x){return 0;} //2
	private int lightsaber (long x){return 0;} //3
	protected long lightsaber(int x){return 0;} //4
	protected long lightsaber (int x, int y){return 0;} //5
	public int lightsaber(int x){return 0;} //6
	protected long lightsaber (long x){return 0;} //7
}
```

2, 3, 5. 6, 7

1 no es debido a que no es un overload correcto, tiene misma firma pero el modificador de acceso es mas restrictivo que el método original, causando error de compilacion

4 no es debido a que no se anda haciendo override correctamente, debido a que el valor de retorno es diferente

3.- Que resultado arroja?

```
class Mouse{
	public int numTeeth;
	public int numWhiskers;
	public int weight;
	public Mouse(int weight){
		this(weight,16);
	}
	public Mouse(int weight, int numTeeth){
		this(weight, numTeeth,6);
	}
	public Mouse (int weight, int numTeeth, int numWhiskers){
		this.weight=weight;
		this.numTeeth=numTeeth;
		this.numWhiskers=numWhiskers;
	}
	public void print(){
		System.out.println(weight+” “+numTeeth+” “+numWhiskers);
	}
	public static void main(String[] args){
		Mouse mouse = new Mouse(15);
		mouse.print();
	}
	}
```

entra al constructor de un solo parámetro, el cual manda llamar al de dos variables mandando su peso cómo 16, el cual a su vez manda llamar al de tres variables mandando su numero de bigotes cómo 6, haciendo que imprima 15, 16, 6

4.- cual es la salida?

```
class Arachnid{
	public String type = “a”;
	public Arachnid(){
```

```
                System.out.println(“arachnid”);
        }
}
class Spider extends Arachnid{
        public Spider(){
                System.out.println(“spider”);
        }
        void run(){
                type=”s”;
                System.out.println(this.type+” “+super.type);
        }
        public static void main (String[] args){
                new Spider().run();
        }
}
```
arachnid spider s s

5.- cual es el resultado?

```
class Test{
        public static void main(String[] args){
                int b=4;
                b--;
                System.out.println(--b);
                System.out.println(b);
        }
}
class Sheep{
        public static void main(String[] args){
                int ov=999;
                ov--;
                System.out.println(--ov);
                System.out.println(ov);
        }
}
```

en ambos casos pasa lo mismo pero son diferentes valores, en el de test, imprime 2 veces 2 debido a que quita una vez con el b-- y luego quita antes de que se muestre el valor con el --b, en el caso de sheep imprime 2 veces 997

6.- Resultado de

```
class Overloading{
        public static void main(String[] args){
                System.out.println(overload(“a”));
                System.out.println(overload(“a”,”b”));
                System.out.println(overload(“a”,”b”,”c”));
        }
        public static String overload (String s){
                return “1”;
        }
        public static String overload(String… s){
```

```
            return “2”;
        }
        public static String overload(Object o){
            return “3”;
        }
        public static String overload(String s, String t){
            return “4”;
        }
}
```
resultado 1, 4, 2
esto porque el string toma prioridad sobre el object

7.-Resultado
```
class Base1 extends Base{
        public void test(){
            System.out.println(“base1”);
        }
}
class Base2 extends Base{
        public void test(){
            System.out.println(“Base2”);
        }
}
class Test{
        public static void main (String[] args){
            Base obj=new Base1();
            ((Base2)obj).test();
        }
}
```
Resultado, ClassCastException, aunque el casteo esta bien hecho, no se puede castear a una clase con la que no tenga relacion alguna.

8.- Resultado
```
public class Fish{
        public static void main(String[] args){
            int numFish=4;
            String fishType=”Tuna”;
            String anotherFish=numFish+1;
            System.out.println(anotherFish+” “+fishType);
            System.out.println(numFish+” ”+1);
        }
}
```
resultado, no compila debido a que se intenta declarar un string con valores enteros

9.- Resultado
```
class MathFun{
        public static void main(String[] args){
```

```
                int number1= 0b0111;
                int number 2= 0111_000;
                System.out.println(“number1: “+number1);
                System.out.println(“number2: “+number1);
        }
}
```
resultado, 7 7, 0111 en binario es 7, y el código imprime dos veces number 1

10.- Resultado
```
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();
        }
}
```
resultado, 20, en calc se sobreescribe el valor por el valor de num*10 siendo num el parámetro dado.

11.-Que aseveraciones son correctas?
```
class ImportExample{
        public static void main(String[] args){
                Random r=new Random();
                System.out.println(r.nextInt(10));
        }
}
```
A) if you omit java.util import statements java compiles gives you an error
B) java.lang and util.random are redundant
C) you dont need to import java.lang

c, java.lang no es necesaria de importar ya que el compilador siempre la importa

12.- resultado
```
public class Main{
        public static void main(String[] args){
                int var=10;
                System.out.println(var++);
                System.out.println(++var);
        }
}
```
resultado, imprime 10 y 12 en ese orden, post es hecho despues de acceder a la info de la variable, pre es hecho antes de acceder

13.- resultado

```
class MyTime{
	public static void main(String[] args){
		short mn=11;
		short hr;
		short sg=0;
		for(hr=mn;hr>6;hr-=1){
			sg++;
		}
		System.out.println(“sg=”+sg);
	}
}
```

imprime sg=5

14.- cuales son verdad

A) an arraylist is mutable
B) an array has a fixed size
C) an array is mutable
D) an array allows multiple dimensions
E) an arraylist is ordered
F) an array is ordered

todas son verdad, aunque e y f son ambiguas, si se encuentran ordenados pero en el orden de entrada, no en un orden especifico, tambien la c es ambigua, si se refiere a sus elementos esta bien, si se refiere a que el array cambia de tamaño no esta bien

15.- Resultado

```
public class MultiverseLoop{
	public static void main(String[] args){
		int negotiate=9;
		do{
			System.out.println(negotiate);
		}while(--negotiate);
	}
}
```

Error de compilacion, while solo acepta booleanos o statements que devuelvan booleanos

16.- resultado

```
class App{
	public static void main(String[] args){
		Stream<Integer> nums=Stream.of(1,2,3,4,5);
		nums.filter(n->n%2==1);
		nums.forEach(p->System.out.println(p));
	}
}
```

resultado, illegalStateException, esto debido a que se intenta acceder dos veces al stream, no se puede esto, por eso se concatenan, si fuera nums.filter(n->n%2==1).forEach(p->System.out.println(p)); funcionaria sin problemas

17.- suppose the declared type of x is a class, and the declared type of y is an interface. When is the assignment x=y; legal?
cuando x es un objeto, asi podria ser asignado por polimorfismo.

18.- when a byte is added to a char, what is the type of the result?
int, cuando se hace cualquier tipo de operacion entre byte, long y char se vuelven int

19.- the standard application programming interface for accessing databases in java is?
la JDBC, java database controller.

20.- which one of the following statements is true about using packages to organize your code in java?
los packages permiten limitar el acceso a clases, métodos o datos a clases que esten fuera del package

21.- forma correcta de inicializar un booleano?

boolean a=true; o boolean a=(n<y);

23.- pregunta

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

resultado, lanza RuntimeException

24.- resultado

```
interface Interviewer{
	abstract int interviewConducted();
	}
public class Manager implements Interviewer{
	int interviewConducted(){
		return 0;
	}
```

```
}
```
compilation error, el método implementado en manager necesita ser public, ya que las clases de una interface son publicas

25.- pregunta

```
class Arthropod{
	public void printName(double Input){
		System.out.println(“Arth”);
	}
}
class Spider extends Arthropod{
	public void printName(int input){
		System.out.println(“Spider”);
	}
	public static void main(String[] args){
		Spider spider=new Spider();
		spider.printName(4);
		spider.printName(9.0);
	}
}
```
imprime Spider Arth

26.- pregunta

```
public class Main{
	public enum Days{Mon,Tue,Wed}
	public static void main(String[] args){
		for (Days d: days.values()){
			Days[] d2=Days.values();
			System.out.println(d2[2]);
		}
	}
}
```
resultado, wed wed wed, siempre imprime el ultimo valor del enum

27.-

```
public class Main{
	public enum Days{MON,TUE,WED}
	public static void main(String[] args){
		boolean x=true,z=true;
		int y=20;
		x=(y!=10)^(z=false);
		System.out.println(x+” “+y+” “+z);
	}
}
```
la declaracion de x devuelve true, porque la segunda parte devuelve el false que fue declarado, el valor de y no fue cambiado, al igual que el de z una vez fue declarado en la operacion de x. imprimiendo true 20 false

28.- pregunta

```
class InitializationOrder{
```

```
	static{add(2);}
	static void add(int num){
		System.out.println(num+" ");
	}
	InitializationOrder(){add(5);}
	static{add(4);}
	{add(6);}
	static {new InitializationOrder();}
	{add(8);}
	public static void main(String[] args){}
}
```

2, 4, 6, 8, 5

esto porque primero se ejecutan los statics, el cual crea una instancia de la clase, haciendo que todos los bloques de código que quedaran se ejecutaran, terminanod con el valor del constructor default

29.- pregunta

```
public class Main{
	public static void main(String[] args){
		String message1="Wham bam");
		String message2=new String("Wham bam");
		if(message1!=message2){
			System.out.println("they dont match");
		}else{
			System.out.println("they match");
		}
	}
}
```

imprime "they dont match", debido a que no son el mismo objeto ni message 2 esta en la pool de strings

30.- pregunta

```
class Mouse{
	public String name;
	public void run(){
		System.out.println("1");
		try{
			System,out.println("2");
			name.toString();
			System.out.println("3");
		}catch(NullPointerException e){
			System.out.println("4");
			throw e;
			}
			System.out.println("5");
	}
	public static void main(String[] args){
		Mouse jerry = new Mouse();
		jerry.run();
```

```
                System.out.println("6");
        }
}
```
imprime 1,2,4 NullPointerException, no toca 3 porque name.toString() suelta la excepcion

31.-pregunta
```
public class Main{
        public static void main(String[] args){
                try(Connection con = driverManager.getConnection(url,uname,pwd)){
                        Statement stmt= con.createStatement();
                        System.out.print(stmt.executeUpdate("INSERT INTO User VALUES
        (500,'ramesh')"));
                }
        }
}
```
imprime 1

32.- pregunta
```
class MarvelClass{
        public static void main(String[] args){
                MarvelClass ab1,ab2,ab3;
                ab1=new MarvelClass();
                ab2=new MarvelMovieA();
                ab3=new MarvelMovieB();
                System.out.println("the profits
are"+ab1.getHash()+","+ab2.getHash()+","+ab3.getHash());
        }
        public int getHash(){
                return 676000;
        }
}
class MarvelMovieA extends MarvelClass{
        public int getHash(){
        return 18330000;
        }
}
class MarvelMovieB extends MarvelClass{
        public int getHash(){
        return 27980000;
}
}
```
imprime 676000, 18330000, 27980000

33.- pregunta
```
class Song{
        public static void main(String[] args){
                String[] arr={"duhast", "feel", "yellow", "fix you"};
                for (int i=0; i<=arr.length(); i++){
```

```
                        System.out.println(arr[i]);
                }
        }
}
```
imprime todo el array para luego mandar excepcion por ArrayIndexOutOfBounds

34.-pregunta
```
class Menu{
        public static void main(String[] args){
                String[] breakfast = {“beans”,”egg”,”ham”,”juice”};
                for(String rs: breakfast){
                        int dish=2;
                        while(dish<breakfast.length){
                                System.out.println(rs+”,”+dish);
                                dish++;
                        }
                }
        }
}
```
imprime 2 beans, 3 beans, 2 egg, 3 egg, 2 ham, 3 ham, 2 juice, 3 juice

35.- which of the following statement are true?

A) string builder es generalmente mas rapido que string buffer
B) string buffer is threadsafe, stringbuilder is not

ambas son verdaderas

36.-
```
class CustomKeys{
        Integer key;
        CustomKeys(Integer k){
                key=k;
        }
        public boolean equals(object o){
                return((CustomKeys)o).key==this.key;
        }
}
```
funciona bien, aunque podria ser no optima la implementacion

37.- pregunta
the catch clause is of the type:

a) throwable
b) exception but not including runtimeException
c) checkedException
d) RunTimeException
e) Error

respuesta, b

38.- an enhanced for loop

is also called for each, offers a simple syntax to iterate through a collection but it cant be used to delete elements of a collection

39.-which of the following methods may appear in class y, which extends x?
public void doSomething(int a, int b){}

40.- pregunta

```
public class Main{
	public static void main(String[] args){
	String s1=”Java”;
	String s2=”java”;
	if(s1.equalsIgnoreCase(s2)){
		System.out.println(“Equal”);
	}else{
		System.out.println(“Not equal”);
	}
}
}
```

imprime equal, el equalsIgnoreCase ignora el que esten en mayusculas o minusculas

41.- pregunta

```
class App{
	public static void main(String[] args){
		String[] fruits={“banana”,”apple”,”pears”,”grapes”};
		Arrays.sort(fruits,(a,b)->a.compareTo(b));
		for(String s: fruits){
			System.out.println(“”+s);
		}
	}
}
```

imprime apple, banana, grapes, pears, el compareto solo fue una manera extensa de decir que iba a comparar conforme a lo que se hace ya

42.- pregunta

```
public class Main{
	public static void main(String[] args){
		int[] countsOfMoose = new int[3];
		System.out.println(countsOfMoose[-1]);
	}
}
```

lanza arrayIndexOutOfBoundsException

43.- pregunta

```
class Salmon{
	int count;
	public void Salmon(){
		count=4;
	}
```

```
	public static void main(String[] args){
		Salmon s= new Salmon();
		System.out.println(s.count);
	}
}
```
cómo Salmon es un método y no un constructor, devuelve el valor default de int, que es 0

44.- pregunta
```
class Circuit{
	public static void main(String[] args){
		runlap();
		int c1=c2;
		int c2=v;
	}
	static void runlap(){
		System.out.println(v);
	}
	static int v;
}
```
si se cambia de lugar la declaracion de c1 y c2, el programa correria sin problemas, ya que ahorita se tiene que hay un problema por una variable no declarada (c2) ya que aun no esta declarada cuando se quiere usar

45.- pregunta
```
class Foo{
	public static void main(String[] args){
		int a=10;
		long b=20;
		short c=30;
		System.out.println(++a + b++ * c);
	}
}
```
imprime 611, esto por ser 11+20*30, 11+600, 611

46.- pregunta
```
public class Shop{
	public static void main(String[] args){
		new Shop().go(“welcome”,1);
		new Shop().go(“welcome”,”to”,2);
	}
	public void go(String… y, intx){
		System.out.print(y[y.length-1]+””);
	}
}
```
falla al compilar, el varargs debe ir al final, no puede ir en ningun otro lado

47.- pregunta
```
class Plant{
```

```
	Plant(){
		System.out.println(“plant”);
	}
}
class Tree extends Plant{
	Tree(String type){
		System.out.println(type);
	}
}
class Forest extends Tree{
	Forest(){
		super(“leaves”);
		new Tree(“leaves”);
	}
	public static void main(String[] args){
		new Forest();
	}
}
```

imprime plant, leaves, plant, leaves, esto porque el constructor de mayor jerarquia es el que termina de ejecutarse primero

48.- pregunta

```
class Test{
	public static void main(String[] args){
		String s1 = “hello”;
		String s2 = new String(“hello”);
		s2=s2.intern();
		System.out.println(s1==s2);
	}
}
```

imprime true, ya que intern devuelve el lugar de memoria de el string que comparte lugar, en este caso, devolveria la misma direccion de memoria que hello haciendolos iguales

49.-cual de las siguientes construcciones es un ciclo infinito while:

a) while(true);
b) while(1==1){}

ambas son verdaderas

50.-

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

el código comentado se podria ejecutar sin problemas, esto debido a que sampleClass es el padre de anothersampleclass

50-2.-pregunta

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

imprime 0, no entro a ninguno de los ifs

51.-pregunta

```
public class Main{
	public static void main(String[] args){
		course c=new course();
		c.name=”java”;
		System.out.println(c.name);
	}
}
class course{
	String name;
	course(){
		course c= new course();
		c.name=”oracle”;
	}
}
```

al correr se crea un bucle de creacion de clases debido a cómo se ejecuta el constructor, llenando eventualmente el stack de clases causando un StackOverflowError

52.-pregunta

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

no compila debido a que no se tiene un valor en a antes de llamar al método toString()

53.- pregunta

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

```
                System.out.println(2+3+5);
                System.out.println("+"+2+3+5);
        }
}
```
imprime 10 y +235

54.-
```
public class Main{
        public static void main(String[] args){
                int a=2;
                int b=2;
                if(a==b)
                        System.out.println("here1");
                if(a!=b)
                        System.out.println("here2");
                if(a>=b)
                        System.out.println("here3");
        }
}
```
imprime here1, here3

55.- pregunta
```
public class Main extends count{
        public static void main(String[] args){
                int a=7;
                System.out.println(count(a,6));
        }
}
class count{
        int count(int x, int y){return x+y;}
}
```
falla al compilar por no pasar los valores que pide el método

56.- pregunta
```
class trips{
        void main(){
                System.out.println("mountain");
        }
        static void main(String args){
                System.out.println("beach");
        }
        public static void main(String[] args){
                System.out.println("magic town");
        }
        void main(Object[] args){
                System.out.println("city");
        }
}
```

imprime magic town, debido a que ese es el main que se ejecuta cuando se manda llamar a la clase de forma predeterminada

57.-pregunta

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

imprime 2, 1

58.- pregunta

```
public class Main{
	public static void main(String[] args){
		List<E> p=new ArrayList<>();
		p.add(2);
		p.add(1);
		p.add(7);
		p.add(4);
	}
}
```

no compila, no se sabe que es e al momento de construir ya que no es un tipo de objeto

59.- pregunta

```
public class Car{
	private void accelerate(){
		System.out.println(“car accelerating”);
	}
	private void break(){
		System.out.println(“car breaking”);
	}
	public void contron(boolean faster){
		if (faster==true)
			accelerate();
		else
			break();
	}
	public static void main (String[] args){
		Car car=new Car();
		car.control(false);
	}
}
```

no compila porque se tiene una palabra reservada en el nombre de un método

60.- pregunta

```
class App{
```

```
	App(){
		System.out.println(“1”);
	}
	App(Integer num){
		System.out.println(“3”);
	}
	App(Object num){
		System.out.println(“4”);
	}
	App(int num1, int num2, int num3){
		System.out.println(“5”);
	}
	public static void main(String[] args){
		new App(100);
		new App(100L);
	}
}
```

imprime 3 y 4, esto porque toma el 100 cómo objeto Integer, mientras que el 100 long lo toma cómo objeto debido a que es lo mejor que se acomoda a este.

61.- pregunta

```
class App{
	public static void main(String[] argos){
		int i=42;
		String s=(i<40)?”life”:(i>50)?”universe”:”everything”;
		System.out.println(s);
	}
}
```

imprime everything, life se imprimiria si fuera menor de 40, cómo no lo es entra al segundo ternario que es i>50, cómo no es imprime everything

62.- pregunta

```
class App{
	App(){
		System.out.println(“1”);
	}
	App(int num){
		System.out.println(“2”);
	}
	App(Integer num){
		System.out.println(“3”);
	}
	App(Object num){
		System.out.println(“4”);
	}
	public static void main (String[] argos){
		String[] sa={“333.6789”,”234.111”};
		NumberFormat inf= NumberFormat,getInstance();
```

```
                inf.setMaximumFractionDigits(2);
                for (String s:sa){
                        System.out.println(inf.parse(s));
                }
        }
}
```
falta el throws parseException o el atrapar esta excepcion para que se compile

63.- Pregunta
```
class Y{
        public static void main(String[] argos){
                String s1=”ocajp”;
                String s2=”ocajp”+””;
                System.out.println(s1==s2);
        }
}
```
imprime true, ya que se le suma una cadena vacia que no altera en nada el valor de la cadena original

64.- pregunta
```
class Y{
        public static void main(String[] argos){
                int score=60;
                switch(score){
                        default:
                                System.out.println(“not valid score”);
                        case score<70:
                                System.out.println(“failed”);
                                break;
                        case score >= 70:
                                System.out.println(“passed”);
                                break;
                }
        }
}
```
case no puede contener un boolean, causando un error de compilacion

65.- pregunta
```
class Y{
        public static void main (String[] argos){
                int a=100;
                System.out.println(-a++);
        }
}
```
imprime -100

65-1.- Which of the following is not a valid array declaration?
int arr4[][] = new int[][8]

siempre se debe especificar la primera dimension de la declaracion del array

66.- pregunta

```
class Y{
	public static void main(String[] argos){
		byte var=100;
		switch(var){
			case 100:
				System.out.println(“var is 100”);
				break;
			case 200:
				System.out.println(“var is 200”);
				break;
			default:
				System.out.println(“default”);
		}
	}
}
```

no se puede compilar debido a que var no es parte de una de las variables admitidas dentro de un switch, solo se puede long, int y short, byte no se puede al igual que los puntos flotantes.

66-1.- which of the following array declarations and initializations is not legal

A) int [] arr3= new int[3]{10,20,30};
B) byte[] val=new byte[10];
C) int[] arr2={1,2,3,4,5};
D) char[] arr1[]=new char[5][];

a, no se puede especificar el tamaño una vez que inicias con datos

67.- pregunta

```
class Y{
	public static void main(String[] argos){
		A obj1=new A();
		B obj2=new B();
		obj2.print();
	}
}
class A{
	public void print(){
		System.out.println(“A”);
	}
}
class B extends A{
	public void print(){
		System.out.println(“B”);
	}
}
```

imprime error de class cast exception, ya que no se puede castear a un padre cómo hijo, a un hijo cómo padre si

68.-pregunta

```
class Y{
	public static void main(String[] argos){
		String fruit = “mango”;
		switch(fruit){
			default:
				System.out.println(“any fruit will do”);
			case “Apple”:
				System.out.println(“apple”);
			case “Mango”:
				System.out.println(“Mango”);
			case “Banana”:
				System.out.println(“banana”);
				break;
		}
	}
}
```

imprime Any fruit will do, apple, mango, banana

69.-pregunta

```
abstract class Animal{
	private String name;
	Animal(String name){
		this.name=name;
	}
	public String getName(){
		return name;
	}
}
class Dog extends animal{
	private String breed;
	Dog(String breed){
		this.breed=breed;
	}
	Dog(String name, String breed){
		super(name);
		this.breed=breed;
	}
	public String getBreed(){
		return breed;
	}
}
class Test{
	public static void main(String[] argos){
		Dog dog1 =new Dog(“beagle”);
```

```
                Dog dog2= new Dog(“Bubbly”, “poodle”);
                System.out.println(dog1.getName() + “:” + dog1.getBreed()+”:”
                        +dog2.getName()+”:”+dog2.getBreed());
}
}
```
falla la compilacion, al no haber una forma de mandar llamar animal con un constructor default, compilaria si se pusiera un super en el constructor de dog.

70.- pregunta
```
public class Main{
        public static void main(String[] argos) throws ParseException{
                String[] sa={“333.6789”,”234.111”};
                NumberFormat nf= NumberFormat.getInstance();
                nf.setMaximumFractionDigits(2);
                for (String s: sa){
                        System.out.println(nf.parse(s));
                }
        }
}
```
imprime 333.6789 u 234.111, no funciona el setMaximumFractionDigits porque no se formatea, solo se parsea para que sea un objeto numero

71.- Pregunta
```
public class Main{
        public static void main(String[] argos) throws ParseException{
                Queue<String> products=new  ArrayDeque<String>();
                products.add(“p1”);
                products.add(“p2”);
                products.add(“p3”);
                System.out.println(products.peek());
                System.out.println(products.poll());
                System.out.println(“”);
                products.forEach(s->{
                        System.out.println(s);
                        });
        }
}
```
imprime p1 p1 y luego p2 p3
esto porque peek solo checa el inicio de la cola, poll quita y devuelve el inicio de la cola, dejando solamente p2 y p3 dentro

72.-
```
public class Main{
        public static void main(String[] argos) throws ParseException{
                System.out.println(2+3+5);
                System.out.println(“+”+2+3+5);
        }
}
```

imprime 10 y +235, esto porque suma strings en vez de valores en la segunda, por la string “+” que hace que se castee el resto cómo string