Unidad 3
Programación Orientada a Objetos. Programación JAVA. Parte I (7.5)

Nota: 85% [Parte I (_____/7.5p) + Parte II (_____/2.5p)] + 15%[Parte III WEB (_____)]
En la parte I se podría obtener un 4 pero como máximo se considera un 2.5p.

Alumno: _____

1. Corregir los errores del siguiente código JAVA. (0.5p):

```
// Obtener el menor divisible por n proporcionado por el usuario
public class Ejercicio1 {
        public static void main(String[] args) throws IOException  {
                int vector[];
                int numero=0;
                int mayor= Integer.MIN_VALUE;
                int menor= Integer.MAX_VALUE;
                BufferedReader in=new BufferedReader(new
                InputStreamReader(System.in));
                System.out.println("¿Dime la longitus del vector?");
                int longitud=Integer.valueOf(in.readLine()).intValue();
                System.out.println("¿Dime el n divisible?");
                numero=Integer.valueOf(in.readLine());
                vector=new int[longitud];
                for(int i=0 ; i<vector.length ;)
                {
                        System.out.println(".....:");
                        vector[i] = Integer.valueOf(in.readLine()).intValue();
                        if ((vector[i] % numero) == 0) || (vector[i] < menor))
                                menor= vector[i];
                }
                System.out.println("....: "+mayor);
                System.out.println("....: "+menor);
        }
}
```

2. Corregir los errores del siguiente código JAVA. (0.5p):

```
import java.io.*;
public class Ejercicio2
{
	public static primo() {
		boolean encontrado = false;
		int divisor=2;
		while(divisor < num && num % divisor == 0)
		{
			divisor++;
		}
		return num != divisor?true:false;
	}
	public static void main(String[] args) throws IOException{
		BufferedReader in=new BufferedReader(new
		InputStreamReader(System.in));
		System.out.println("Introduce un numero: ");
		int num = Integer.valueOf(in.readLine().trim()).intValue();
		if (primo(num))
			System.out.println("El numero "+num+" es primo");
		else
			System.out.println("EL numero "+num+" no es primo");
	}
}
```

3. ¿Qué resuelve el método int z(int num)? (0.5p):

```
import java.io.*;
public class Ejercicio6 {
public static int z(int num){
        int y = 1;
        for (; num>1 ; num--) y = y * num;
        return(y);
}
public static void main(String[] args) throws IOException{
        BufferedReader in = new BufferedReader(new
        InputStreamReader(System.in));
        System.out.println("……..: ");
        int num = Integer.valueOf(in.readLine().trim()).intValue();
        System.out.println("…….: "+z(num));
        }
}
```

4. Qué se añadiría para mejorar -robustez- el código que se proporciona:

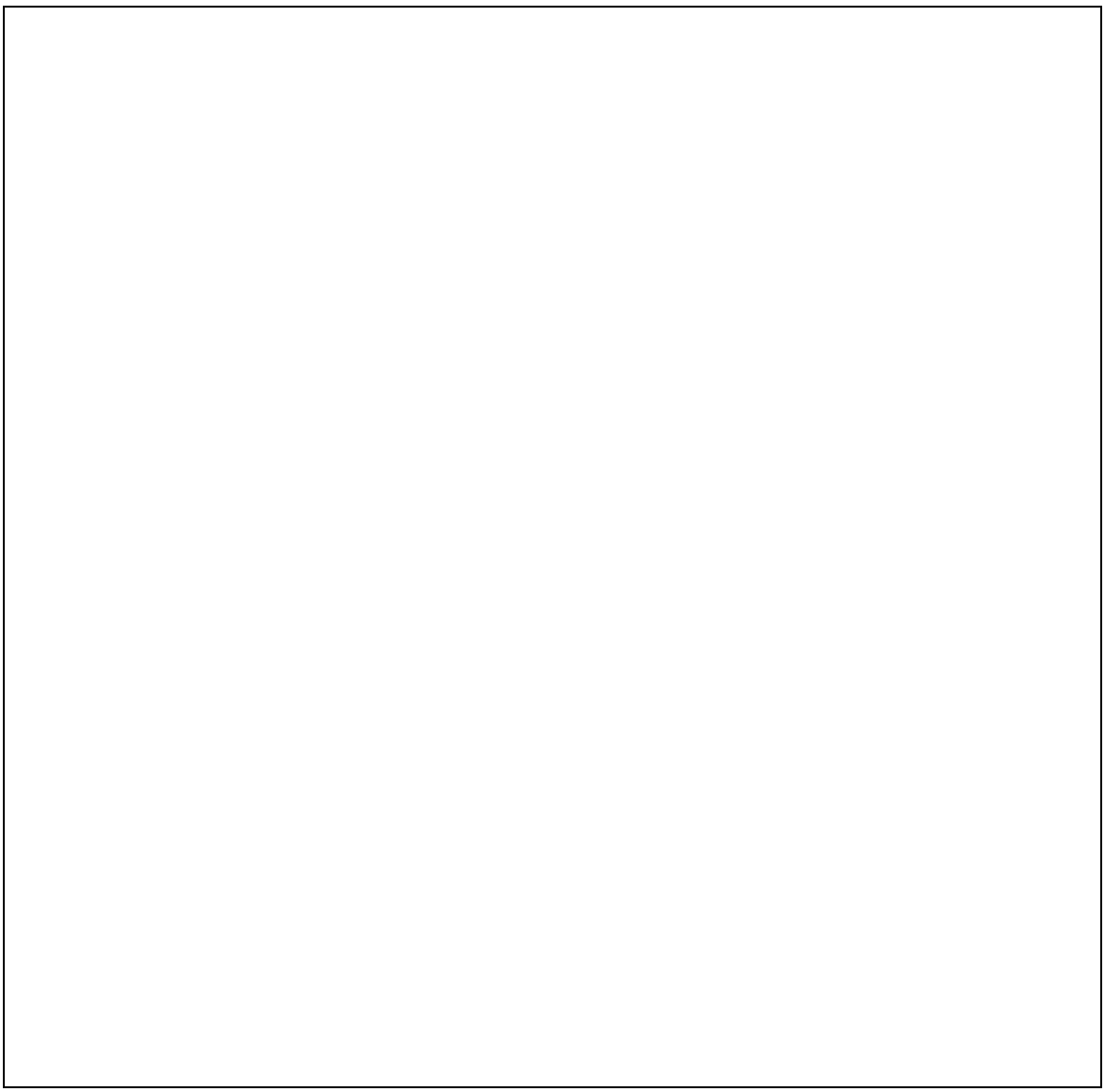
```
class Bombilla
 {
  private ...; // interruptor
  public void enciender () { ... }
  public void apagar () { ... }
  public boolean encendida () { .}
  public Bombilla (){ ... }
  }
```

```
class Iluminacion
{
private ...;  // Bombillas de la casa
private ...;  // Número de bombillas de la casa
public void apagar_bombilla() { ... }
public void encender_bombilla() { ... }
public boolean estado_bombilla() { ...}
public numero_bombillas_encendidas (){ ... }
public numero_bombillas_apagadas (){ ... }
public Iluminación { ... }
}
```

```
Public boolean estado_bombilla (int aux) { // (0.5p)
        return(this.iluminacion[aux].interruptor)
}
```

5. Qué se añadiría para mejorar -robustez- el código que se proporciona (0.5p):

```
// Obtener la media de un vector de enteros
public class Ejercicio4 {
    public static void main(String[] args) throws IOException {
            int vector[] = new int[10]
            int menor= Integer.MAX_VALUE;
            int suma=0;
            BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
            for(int i=0 ; i< vector.lenght; i++)
            {
               System.out.println(".....:");
               vector[i] = Integer.valueOf(in.readLine()).intValue();
               suma+= vector[i];
            }
            System.out.println("....: "+suma/vector[i]);
   }
}
```



6. Qué se añadiría para mejorar -robustez- el código que se proporciona (0.5p):

```
public gpoligono(int num, Rgb c)
{
        /*
        Números de lados (int). int nlados;
        Vector de (class Punto). Punto [] posicionamiento;
        Color de relleno (class Rgb). Rgb color;
        */
        this.nlados=num;
        this.posicionamiento=new Punto[num];
        this.color=c;
}
```

Recordatorio para la creación de Excepciones:

```
public class Error01Exception extends Exception
{
        public Error01Exception(String mensaje)
        {
                super(mensaje);
        }
}
```

Para lanzarlo: ***throw new Ejercicio1Exception("Error, numero nulo detectado");***

7. Escriba en la clase Java Gpoligono el siguiente método (0.5p):

Atributos:

- ✓ Números de lados (*int*). int nlados;
- ✓ Vector de (*class* ***Punto***). Punto [] posicionamiento;
- ✓ Color de relleno (*class* ***Rgb***). Rgb color;

Métodos:

- ✓ public gpoligono(){// Constructor por defecto}
- ✓ public gpoligono(int num, Rgb c){/* Mediante este constructor sobrecargado se proporciona el número de lados y su color. Dentro del cuerpo de este método te preguntará por cada uno de los puntos (X,Y) que formarán su posicionamiento en el espacio.*/}
- ✓ public int getlados(){// Método que devuelve los lados del Gpoligono.}
- ✓ public Rgb getcolor(){// Método que devuelve el color del Gpolígono.}

- ✓ public boolean compareTopoligono(Gpoligono a){/* Método que dado un polígono determina si es igual a él. Son iguales cuando coinciden el color, número de lados y su posicionamiento. */}
- ✓ public Gpoligono mayorTopoligonos(Gpoligono [] a){/* Método que dado un vector de polígonos devuelva el polígono con mayor número de lados. */}

New Método: **public int suma_lados(Gpoligono [] a){/* Método que dado un vector de polígonos devuelva la suma de los lados de cada uno de los polígonos. */}**

```
public int suma_lados(Gpoligono [] a) {




} // Fin del new método
```

8. Implementa los métodos resaltados de la clase CajeroAutomático:

```
public class Operacion
{
        private String nTarjeta; // número de la tarjeta
       public String operación; // tipo de operación por ejemplo VERSALDO
       public int cantidad; // Cantidad monetaria retirada en una operación de RETIRAR
        // Contructor de inicializar cada uno de los atributos de una instrución
        public Operacion (String nt, String op) { ... } // En este caso la cantidad tendrá un
       valor de -1 euros
       public Operacion (String nt, String op, int cantidad) { ... }
       // Devuelve el código de la operación.
        public String getOperacion () {…. }
 }
```

```
public class CajeroAutomatico
{
        public Operacion[] cola; // vector de operaciones
        private int numTransacciones; // número de instrucciones en el dispositivo
       // Constructor tamaño inicial de 100
       public CajeroAutomatico () {
            this.cola=new Operacion[100];
            this. numTransacciones =0;          }
       // Constructor con tamaño inicial proporcionado por el usuario
       public CajeroAutomatico (int tama_max) { …};
        // Método que incorpora una operación al sistema
        public void putITransacciones (Operacion i) { ... }
       // Método que determina cuantas operaciones ha realizado una trajeta
        public int count_operaciones(String nt) { ... }
       // Método que dado una tarjeta determina la cantidad total de dinero retirada.
       public int cantidadRetirado(){….}
}
```

```
public int count_operaciones(String nt) // (0.5p)
{




} // Fin del new método
```

```
public int cantidadRetirado(String nt) // (0.5p)
{




```

} // Fin del new método

9. Escriba *los dos nuevos métodos que se proponen* de la clase Java que represente un **Dispositivo**:

```
public class Instruccion{
        private String codigo; // nombre de la instrucción
        public int prioridad; // prioridad de la instrucción
        public long tiempo; // instante en el que se invoca la instrucción
        // Contructor de inicializar cada uno de los atributos de una instrución
        public Instruccion (String codigo, int prioridad, long tiempo) { ... }
       // Devuelve el código/nombre de la insrtucción. Por ejemplo podría devolver
       "SUM"
        public String getCodigo () {…. }
       // Método que compara dos instrucciones devolviendo true si ambas tienen el
       mismo código/nombre de acción
       public boolean compareToIntruccion (Instrucción aux){…}
}

public class Dispositivo{
        private Instruccion[] cola; // vector de instrucciones
        private int numInstrucciones; // número de instrucciones en el dispositivo
       // Constructor tamaño inicial de 100
       public Dispositivo () {
            this.cola=new Instrucción[100];
            this.numInstrucciones=-1;          }
       // Constructor con tamaño inicial proporcionado por el usuario
       public Dispositivo (int tama_max) { …};
        // Método que se ejecuta cuando se quiere introducir una instrucción
        public void putInstruccion (Instruccion i) { ... }
       // Método que determina cuantas instrucciones son del nombre/código "XX"
        public int count_codigo (String codigo) { ... }
       // Método que ejecuta una instrucción.
```

public void ejecutarInstruccion(){….}
public int pos_mayor prioridad () *{/* Devuelve la posición de la instrucción con mayor prioridad. Si el sistema está vacío devuelve -1}*
public void ejecutarInstruccion(int pos)*{ /*Método que ejecuta la instrucción que ocupa la posición p. La ejecución de la instrucción cola[p] implica simularlo mostrando por pantalla las instrucción y a continuación desplazando todas la siguientes una posición a la izquierda y actualizando el valor de numInstrucciones. Si no hay instrucciones debe aparecer por pantalla "dispositivo vacío" */}*
public Dispositivo mayor_carga (Dispositivo [] aux) *{/*Método que devuelve el dispositivo con mayor número de instrucciones a ejecutar */}*

}

public void exec_MAX_Prioridad(){ // (0.5p)
// Ejecutar la instrucción de máxima prioridad!!!!

} // Fin del new método

public Dispositivo mayor_prioridad (Dispositivo [] aux) { // (0.5p)
// Se devuelve el Dispositivo con instrucción de mayor prioridad

} // Fin del new método

10. Cuestionario tipo test(3 mal quitan 1 bien): (2 puntos)

1) ¿ Qué ocurre cuando se compila y ejecuta la siguiente clase ?

```
class MiClase {
public static void main (String[] args) {
String s1[] = new String[5];
String str = s1[0].toUpperCase();
System.out.println(str);
}
}
```

   a. Imprime NULL.
   b. Da un error al compilar.
   c. Imprime null.
   d. Da una excepción NullPointerException al ejecutar.

2) Qué se escribe por pantalla con la siguiente línea de código: System.out.println ((int) Math.PI);
   a. 3.1415926
   b. 3
   c. nada, hay un error al compilar.
   d. nada, hay un error durante la ejecución.

3) Dada la siguiente definición: ¿Cuál debería ser el contenido del constructor?

```
class Temperatura {
double t;
Temperatura(double t) { ¿? }
}
```

   a. t=t;
   b. double t=t;
   c. this.t=t;
   d. No se puede llamar igual el parámetro del constructor que el atributo de la clase

4) "for (x= 5; x <100; x*= 2) { cosas; }" es equivalente a ...
   a. x= 5; do { cosas; x*= 2; } while (x <100);
   b. x= 5; while (x <100) { cosas; x*= 2; }

5) 2. Dado el siguiente fragmento de programa: Indique que afirmación es cierta:

```
int k;
for (k=5 ; k>0 ; k--)
System.out.print(k);
System.out.print(k);
```

   a. Se imprime 543210
   b. Se imprime 5432100
   c. Se imprime 554433221100
   d. Se imprime 543210-1

6) Indique el elemento que no es obligatorio en una declaración de variable:

   a. La asignación del valor inicial.
   b. El identificador o nombre.
   c. El punto y coma.
   d. El tipo.

7) Indique la salida de:

```
int a= 7, b= 3;
System.out.println ((++a) * b);
```

a. 24
b. 21
c. 10
d. 73

8) Evalúe el valor final que toma la variable "s":

```
int n= 1; s= 0;
while (n <= 9) s+=n;
```

a. 45
b. 0
c. 9
d. el programa no termina nunca

9) ¿ Qué imprime este programa ?

```
int metodo (int v1) { return v1*v1; }
   ……
int v = 3;
System.out.print (metodo (metodo (v))); ...
```

a. Tiene errores que impiden su compilación
b. 9
c. 81
d. Compila; pero tiene errores que impiden su ejecución

10) ¿Qué se imprimirá al ejecutar el siguiente bucle?

```
for (int i=0; i < 5; i++) {
if (i==3) { i=5; }
System.out.println (i + " ");
}
```

a. 0 1 2 3 4
b. 0 1 2 5
c. 0 1 2 4
d. 0 1 2

11) Indique qué ocurre con el siguiente código:

```
int a[] = new int[10];
for (int i=0 ; i<=a.length ; i++)
   System.out.print(a[i]);
```

a. Error de ejecución: índice fuera de rango.
b. Error de compilación: no se asignaron valores al array.
c. Escribe los 10 valores almacenados en a.
d. Escribe 10 ceros.

12) Indique qué escribe una llamada al método m()

```
void m() {
   int x = 0;
```

```
try {
   System.out.print(x++);
   if (x>0) throw new Exception();
   System.out.print(x++);
} catch (Exception e) {
   System.out.print(x++);
   } finally {
   System.out.print(x++);
  }
}
```

a. 012
b. 0
c. 01
d. 0123

13) Dados los siguientes fragmentos de código:

```
class ClaseC {
public void fmet (int i) { ... }
public int fmet (int i) { ... }
}  ...
ClaseC c = new ClaseC();
c.fmet(4);
```

Se produce:

a. la llamada al primer método fmet.
b. la llamada a ningún método porque hay sobrecarga.
c. un error al compilar.
d. un error al ejecutar.

14) Dado el método "imprime":

```
void imprime (boolean ok) {
 System.out.println ("La respuesta " + (ok?"NO":"") + " está mal");
}
```

¿qué imprime la llamada imprime(false)?

a. La respuesta está mal
b. La respuesta NO está mal
c. Se produce un error de compilación

15) En una sentencia try-catch-finally:

a. Los bloques catch se pueden repetir tantas veces como excepciones de distinto tipo se desee atrapar. El bloque finally debe aparecer al menos una vez y se ejecuta siempre.
b. Los bloques catch se pueden repetir tantas veces como excepciones de distinto tipo se desee atrapar. El bloque finally no es opcional y se ejecuta siempre.
c. Los bloques catch se pueden repetir tantas veces como excepciones de distinto tipo se desee atrapar. El bloque finally es opcional y solo puede aparecer una vez. Este bloque se ejecuta siempre.
d. Todas las afirmaciones son falsas

Unidad 3
Programación Orientada a Objetos. Programación JAVA. Parte II (2.5p)

Implementa las Aficiones de un Usuario (1p):

Se va a construir un programa para comprobar el grado de afinidad (las posibilidades de amistad) entre usuarios. Cada usuario se representará mediante un objeto, que contiene su dirección de correo electrónico y un array de 20 números reales entre 0.0 y 1.0, que indican el interés del usuario respecto a 20 aficiones diferentes.

Escriba la clase usuario Usuario con:

Atributos:

public String email;
public double [] aficiones;

Métodos:

void Usuario(String correo) {…} (0.25p)
public void setAficion (double aficion, int indice) {…} (0.25p)
public double getAficion (int indice) {…} (0.25p)
public double grado_afinidad(Usuario a) {…} (0.25p)

Para calcular el grado de afinidad entre dos usuarios se usa la aproximación del coseno (coseno del ángulo que formarían los dos arrays de aficiones considerados como vectores en un espacio de 20 dimensiones), variando entre 0 (coincidencia total o afinidad máxima) y 1 (afinidad mínima). Siendo x i el valor de interés en la afición i para el usuario x, e yi el correspondiente para el otro usuario, el grado de afinidad se calcula así:

$$\cos(\vec{x},\vec{y}) = \frac{\sum_{i=0..19} x_i y_i}{\sqrt{\sum_{i=0..19} x_i^{\,2} \sum_{i=0..19} y_i^{\,2}}}$$

Nota: Opcional

Escriba un método para calcular el usuario medio del array de usuarios que se pasa como parámetro. El usuario medio es uno con dirección de correo vacía, en el que cada casilla de afición contiene el valor medio de las casillas de aficiones correspondientes de los usuarios que se pasan en el parámetro.

public static Usuario getUsuarioMedio (Usuario[] usuarios) {…}

Implemente un router (3p):

Escribe la clase Ip:

Atributos:
public int x: 0..255 // x.168.1.250
public int y: 0..255 // 192.y.1.250
public int w: 0..255 // 192.168.w.250
public int z: 0..255 // 192.168.1.z
Métodos
public Ip (int ax,int by,int cw,int dz) {…}
public boolean compareToIp (Ip aux) {…}

Escribe una clase Paquete que modele el datagrama a nivel ip, con los siguientes campos:

Atributos:
public Ip ip_origen: ip de origen del datagrama
public Ip ip_destino: ip de destino del paquete de información
public int QoS: factor de calidad del servicio -0,1..5-
public int saltosmax: número máximo de saltos o router transitados
Métodos
public Paquete(Ip origen,Ip destino,int calidad,int saltos) {…}
public boolean compareToPaquete(Paquete aux) {…}

Escriba una clase que modele un Router, con los siguien tes campos:

Atributos:

public String marca_route;
public Ip ip_route;
public Paquete [] in: // Buffer de entrada de paquetes
public Paquete [] out: // Buffer de salida de paquetes
public int np_int: //número de paquetes ip que están a la espera de entrar
public int np_out: //número de paquetes ip que están a la espera de salir

La clase Router tendrán las método:

un constructor : se definir un buffer de un buffer in/out con tamaño máximo de 20
public boolean llenoRouterIn() {…}
public boolean vacioRouterIn() {…}
public void put_int(Paquete a) {…}
public Paquete get_int() {…} // Obtiene el in[0] despalzando los siguientes una pos
public int numero_paquetes_Iporigen_in(Ip origen) {…}
public int numero_paquetes_QoS_in(int q) {…}
public int [] numero_paquetes_tipos_QoS_in(){…} // numero de paquetes para cada tipos de valores para QoS 0,1..5

hay 12 métodos a razón de 0.25 cada uno de ellos →