1 - Ejercicio 1 *(2 puntos)*

Implementa una clase ***Trabajador*** con los siguientes atributos y métodos:

- ✓ **Atributos privados**: nombre, edad, categoría, antigüedad.
- ✓ **Constantes static públicas.** Determinan los diferentes tipos de categorías y antigüedad de un empleado disponibles en el sistema:
  - ➢ CAT_EMPLEADO. Valor 0
  - ➢ CAT_ENCARGADO. Valor 1
  - ➢ CAT_DIRECTIVO. Valor 2
  - ➢ ANT_NOVATO. Valor 0
  - ➢ ANT_MADURO. Valor 1
  - ➢ ANT_EXPERTO. Valor 2
- ✓ ***Trabajador(String nombre, int edad, int categoria, int antigüedad)***. Constructor de la clase que inicializa los atributos de la misma. Comprobará que la categoría y la antigüedad sean correctos, si no lo son se lanzará la excepción ***IllegalArgumentException***. ***(0.75 puntos)***
- ✓ Métodos públicos **consulta/cambia** de cada uno de los atributos. Se debe comprobar la validez de los atributos y lanzar la excepción correspondiente si hay algún valor incorrecto. ***(0.5 puntos)***
- ✓ ***public double calcularSueldo()***. Devuelve el sueldo del empleado calculado a partir de su antigüedad y categoría profesional. La forma de calcular el sueldo del empleado será de acuerdo a la siguiente tabla ***(0.75 puntos)***:

Sueldo base	607 €
Empleado **Encargado** **Directivo**	+15% sueldo base +35% sueldo base +60% sueldo base
Novato **Maduro** **Experto**	+150 € +300 € +600 €

RESPUESTA

```
public class Trabajador {

    // constantes públicas estáticas
    public static final int CAT_EMPLEADO = 0;
    public static final int CAT_ENCARGADO = 1;
    public static final int CAT_DIRECTIVO = 2;
    public static final int ANT_NOVATO = 0;
    public static final int ANT_MADURO = 1;
    public static final int ANT_EXPERTO = 2;
    //atributos privados
    private String nombre;
    private int edad;
    private int categoria;
    private int antigüedad;

    /**
     * Crea el objeto Trabajador, si existe algún error en la categoría o la
     * antigüedad se lanza la excepción IllegalArgumentException. Como el
     * enunciado no lo expecifica, no es necesario comprobar el resto de parámetros
     * @param nombre
     * @param edad
     * @param categoria
     * @param antigüedad
     */
    public Trabajador(String nombre, int edad, int categoria, int antigüedad) {
        if (comprobarCategoria(categoria) && comprobarAntigüedad(antigüedad)) {
            this.nombre = nombre;
            this.edad = edad;
```

```
            this.categoria = categoria;
            this.antigüedad = antigüedad;
        }
        else{
           throw new IllegalArgumentException("Categoría o antigüedad incorrectos");
        }
    }

    public int getAntigüedad() {
        return antigüedad;
    }

    public void setAntigüedad(int antigüedad) {
        if(comprobarAntigüedad(antigüedad))
            this.antigüedad = antigüedad;
        else{
            throw new IllegalArgumentException("Antigüedad incorrecta");
        }
    }

    public int getCategoria() {
        return categoria;
    }

    public void setCategoria(int categoria) {
        if(comprobarCategoria(categoria))
            this.categoria = categoria;
        else
            throw new IllegalArgumentException("Categoria incorrecta");
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public double calcularSueldo() {
        double sueldo=0;
        switch(this.categoria){
            case CAT_EMPLEADO:
                sueldo+=sueldo*0.15;
                break;
            case CAT_ENCARGADO:
                sueldo+=sueldo*0.35;
                break;
            case CAT_DIRECTIVO:
                sueldo+=sueldo*0.60;
                break;
        }
        switch(this.antigüedad){
            case ANT_NOVATO:
                sueldo+=150;
                break;
            case ANT_MADURO:
                sueldo+=300;
                break;
            case ANT_EXPERTO:
                sueldo+=600;
                break;
        }
        return sueldo;
    }
    private boolean comprobarCategoria(int categoria){
        boolean check=true;
        if(categoria<CAT_EMPLEADO || categoria>CAT_DIRECTIVO)
            check=false;
```

```
        return check;
    }
    private boolean comprobarAntigüedad(int antigüedad){
        boolean check=true;
        if(antigüedad<ANT_NOVATO || antigüedad>ANT_EXPERTO)
            check=false;
        return check;
    }
}
```

Ejercicio 2 (1.5 puntos)

- ✓ Implementa dos funciones para obtener, la parte entera y la parte decimal de un número en punto flotante (***double***). La definición de las funciones es como sigue:
- ✓ ***int getParteEntera(double numero)*** ***(0.5 puntos)***
- ✓ ***int getParteDecimal(double numero)*** ***(0.5 puntos)***
- ✓ Internamente, estas dos funciones convierten el número ***double*** a cadena de caracteres para obtener la parte correspondiente.
- ✓ Una vez obtenido el valor buscado, la cadena se convertirá al tipo de retorno indicado.

Además el programa principal (***main***) pedirá al usuario que introduzca un número por teclado y posteriormente mostrará un menú por pantalla en el que se pregunte si desea obtener la parte entera o decimal del número introducido. El programa principal ***main*** se ejecutará hasta que el usuario introduzca la opción adecuada para salir ***(0.5 puntos).***

RESPUESTA

```
import java.util.Scanner;

/**
 *
 * @author jose
 */
public class Ejercicio2 {

    public static int getParteEntera(double numero){
        String snumero = String.valueOf(numero);
        int posicion_coma = snumero.indexOf('.');
        int parte_entera = Integer.parseInt(snumero.substring(0, posicion_coma));
        return parte_entera;
    }
    public static int getParteDecimal(double numero){
        String snumero = String.valueOf(numero);
        int posicion_coma = snumero.indexOf('.');
        int parte_decimal = Integer.parseInt(snumero.substring(posicion_coma+1));
        return parte_decimal;
    }
    public static void main(String [] args){
        Scanner entrada_teclado = new Scanner(System.in);
        int opcion_menu=0;
        double numero=0;
        do{
            //lectura de un valor Double válido
            boolean numero_valido=false;
            do{
              System.out.println("Introduzca un número double válido: ");
              if(entrada_teclado.hasNextDouble()){
                  numero = entrada_teclado.nextDouble();
                  numero_valido=true;
              }
              else{
                  System.out.println("Número erroneo introducido. Vuelva a intentarlo");
                  entrada_teclado.next();
              }
            }while(!numero_valido);
            //mostramos el menú de opciones
            System.out.println("¿Qué desea hacer con el número?");
```

```
            System.out.println("1.Obtener la parte entera.");
            System.out.println("2.Obtener la parte decimal");
            System.out.println("3.Salir del programa");
            //leemos una opción válida
            numero_valido=false;
            do{
              System.out.println("Introduzca opción: ");
              if(entrada_teclado.hasNextInt()){
                  opcion_menu = entrada_teclado.nextInt();
                  if(opcion_menu>=1 && opcion_menu<=3)
                      numero_valido=true;
                  else
                      System.out.println("Debe introducir un valor entre 1 y 3");
              }
              else{
                  System.out.println("Número erroneo introducido. Vuelva a intentarlo");
                  entrada_teclado.next();
              }
            }while(!numero_valido);
            //Según opción introducida hacemos la acción deseada
            switch(opcion_menu){
                case 1:
                    System.out.println("La      parte      entera      de      "+numero+"      es
"+getParteEntera(numero));
                    break;
                case 2:
                    System.out.println("La      parte      decimal      de      "+numero+"      es
"+getParteDecimal(numero));
                    break;

            }
        }while(opcion_menu!=3);
    }
}
```

Ejercicio 3 *(2 puntos)*

Realizaremos un programa que calcule el valor de los **coeficientes binomiales**. Los coeficientes binomiales son ampliamente utilizados en distintas ramas de las matemáticas, como por ejemplo la estadística. Se representan como un binomio (dos valores) entre paréntesis (similar a una fracción pero sin la barra horizontal). La fórmula para calcular el coeficiente binomial de dos valores **n** y **k**, lo representaremos como **(n,k)** es la siguiente:

$$(n, k) = \frac{n!}{k!\,(n-k)!}$$

donde **!** representa el factorial y **n** y **k** mayores o iguales que **0**.

Para realizar este ejercicio sigue los siguientes pasos:

a) Se implementará una función que calcule el factorial de un número. El número se pasará por parámetro y retornará el factorial calculado. La declaración de la función será como sigue:

(0.75 puntos)

public static int factorial(int n)

El factorial de un número **n**, **n!** Se calcula de la siguiente manera:

- 1, si n=0.
- n*n-1....*1, si n>0.

Por ejemplo, el factorial de **3**, `3! = 3*2*1 = 6`.

b) Se implementará una función que calcule el coeficiente binomial. Los valores de **n** y **k** se pasarán por parámetro y la función devolverá el coeficiente calculado. Se utilizará la función ***factorial()*** para realizar este cálculo. La declaración de la función será como sigue ***(0.5 puntos)***:

public static int coeficienteBinomial(int n, int k)

c) El programa principal (***main***) deberá pedir al usuario que introduzca los valores de **n** y **k** (mayor o igual que 0). Se calculará el valor del binomio llamando a la función **coeficienteBinomial()** y se mostrarán los resultado obtenidos por pantalla. ***(0.75 puntos)***.

RESPUESTA

```
import java.util.Scanner;

/**
 *
 * @author jose
 */
public class Ejercicio3 {

    public static int factorial(int n) {
        int fact = 1;
        int contador = 2;
        while (contador <= n) {
            fact *= contador++;
        }
        return fact;
    }

    public static int coeficienteBinomial(int n, int k) {
        return factorial(n) / (factorial(k) * factorial(n - k));
    }

    public static void main(String[] args) {
        Scanner entrada_teclado = new Scanner(System.in);
        boolean numero_valido = false;
        int n = 0, k = 0;
        do {
            System.out.println("Introduzca N: ");
            if (entrada_teclado.hasNextInt()) {
                n = entrada_teclado.nextInt();
                if (n>0) {
                    numero_valido = true;
                } else {
                    System.out.println("Debe introducir un valor mayor que 0");
                }
            } else {
                System.out.println("Número erroneo introducido. Vuelva a intentarlo");
                entrada_teclado.next();
            }
        } while (!numero_valido);
        do {
            System.out.println("Introduzca K: ");
            if (entrada_teclado.hasNextInt()) {
                k = entrada_teclado.nextInt();
                if (n>0) {
                    numero_valido = true;
                } else {
                    System.out.println("Debe introducir un valor mayor que 0");
                }
            } else {
                System.out.println("Número erroneo introducido. Vuelva a intentarlo");
                entrada_teclado.next();
            }
        } while (!numero_valido);
        System.out.println("El coeficiente binomial (n,k)="+coeficienteBinomial(n,k));
    }
}
```

Ejercicio 4 (0.75 puntos)

Implementa un algoritmo en el que dado un entero n > 1 leído por teclado, calcule e imprima los elementos correspondientes a la **Conjetura de Ullman** (en honor al matemático *S. Ullman*). La conjetura consiste en lo siguiente:

- ✓ Empieza con cualquier entero positivo.
- ✓ Si es par, se divide entre 2; si es impar se multiplica por 3 se agrega 1.
- ✓ Se itera hasta obtener el número 1.

Al final se obtendrá el número 1, independientemente del entero inicial. Por ejemplo, cuando el entero inicial es 26, la secuencia será:

26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

RESPUESTA

```
import java.util.Scanner;

/**
 *
 * @author jose
 */
public class Ejercicio4 {
    public static void main(String [] args){
        Scanner entrada_teclado = new Scanner(System.in);
        boolean numero_valido = false;
        int n = 0;
        do {
            System.out.println("Introduzca un número positivo: ");
            if (entrada_teclado.hasNextInt()) {
                n = entrada_teclado.nextInt();
                if (n>0) {
                    numero_valido = true;
                } else {
                    System.out.println("Debe introducir un valor mayor que 0");
                }
            } else {
                System.out.println("Número erroneo introducido. Vuelva a intentarlo");
                entrada_teclado.next();
            }
        } while (!numero_valido);
        //algoritmo para la conjetura
        System.out.print(n+", ");
        while(n!=1){
            if(n%2==0)
                n/=2;
            else
                n=n*3+1;
           System.out.print(n+", ");
        }
    }
}
```

Ejercicio 5 (0.75 puntos)

Implemente una función que sirva para cifrar un texto con el conocido método de César. El criptosistema consiste en el desplazamiento de 3 caracteres en la posición del carácter a cifrar, es decir, la A se sustituye por la D, la B por la E, …, la X por la A, la Y por la B y la Z por la C. Por simplicidad, supondremos que el texto a cifrar solo contiene caracteres alfabéticos. Por tanto el ejercicio consiste en implementar la siguiente función:

public String cifradoCesar(String cadenaACifrar)

La función recibe como parámetro la cadena a cifrar y devuelve un objeto String con la cadena cifrada mediante el sistema de Cesar.

```
public class Ejercicio5 {
    public static String cifradoCesar(String cadenaACifrar){
        String abc="ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
        int pos_c;
        String cadena_cifrada="";
        for(int i=0;i<cadenaACifrar.length();++i){
            //obtenemos la posición del caracter a cifrar en el abcedario
```

```
            pos_c=abc.indexOf(cadenaACifrar.charAt(i));
            //obtenemos el caracter tres posiciones avanzado, cuidando el extremo
            cadena_cifrada+=abc.charAt((pos_c+3)%abc.length());
        }
        return cadena_cifrada;
    }
    public static void main(String [] args){
        String abc="ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
        System.out.println("Cifrado Cesar: "+cifradoCesar(abc));
    }
}
```

Módulo: PROGRAMACIÓN
I.E.S.: AGUADULCE **C.F.G.S. : DAM y DAW**
Fecha: 2 DE FEBRERO DE 2012 **Hora de Comienzo: 15:30**
Examen: PARTE 1 (TEORÍCA) **Duración: 1:00 h**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

INSTRUCCIONES:

Marca el profesor que tengas asignado: **Fran [ ]** **José Luis [ ]**

La puntuación total del examen PARTE 1 + PARTE 2 será de 10 puntos. Parte teórica: 3 puntos**. Parte Práctica:** 7 puntos.

- **La nota del examen se calculará como la suma de la parte teórica (3 puntos) y la parte práctica (7 puntos), siempre y cuando la nota de la parte teórica sea mayor o igual que 1 y la nota de la parte práctica sea mayor o igual que 2,5.**

- **Para el examen práctico se podrá hacer uso del material bibliográfico que se estime oportuno, así como de apuntes. No obstante, se advierte del peligro de pérdida de tiempo que conlleva ponerse a consultarlo durante el examen, pudiendo consumirse el tiempo disponible en la consulta, y quedándose sin tiempo para las respuestas.**

CUESTIONES TEÓRICAS: (3 puntos) (Cada pregunta correcta puntuará con 0,1 puntos, cada incorrecta restará 0,05 puntos. Si de deja sin contestar ni suma ni resta)

1. ¿Qué orden debemos introducir en línea de comandos para poder obtener un archivo .class, si tuviéramos un archivo llamado programa.java?
a. javac.exe
b. javac programa.java
c. java programa.java
d. ./java programa.class

2. Las aplicaciones Java creadas para su ejecución en dispositivos simples o dispositivos móviles son...
a. Aplicaciones de consola.
b. Servlets.
c. Applets.
d. Midlets.

3. ¿En cuál de las fases de la programación se realiza la compilación del programa?
a. En la fase de resolución del problema.
b. En la fase de implementación.
c. En la fase de explotación.
d. En la fase de mantenimiento.

4. El conjunto finito de símbolos y palabras especiales, es a lo que llamamos:
a. Sintaxis del lenguaje de programación.
b. Léxico del lenguaje de programación.
c. Gramática del lenguaje de programación.
d. Semántica del lenguaje de programación.

5. El lenguaje máquina es directamente interpretable por:
a. El compilador.
b. Un circuito microprogramable.
c. La memoria RAM.
d. Las personas.

6. Cuando se oculta la información para poder implementarla de diferentes maneras sin que esto influya en el resto de elementos, decimos que estamos aplicando...
a. Abstracción.
b. Encapsulación.
c. Corrección.
d. Algoritmos.

7. Cuando los pasos que permiten resolver un problema están escritos en algún lenguaje de programación, estamos hablando de...

a. Algoritmos.
b. Programas.
c. Algoritmos y programas.
d. Lenguajes de programación.

8. El diseño descendente, también recibe el nombre de:

a. Diseño de algoritmos.
b. Diseño modular.
c. Top-up design.
d. Up-down design.

9. Indica los valores de x y z después de las siguientes sentencias:

```
int x = 10;
int z = x++%5;
```

a. x es 9 y z es 1.
b. x es 11 y z es 0.
c. x es 9 y z es 0.
d. x es 11 y z es 1.

10. Respecto a los literales para tipos de dato en coma flotante podemos afirmar que…

a. Los definidos como float usan para su representación un espacio de 32 bits.
b. Los definidos como float usan para su representación un espacio de 4 bytes.
c. Los definidos como double usan para su representación un espacio de 8 bytes.
d. Todas las anteriores son ciertas.

11. El operador que utilizamos para invertir el valor de un boolean es

a. \~
b. !
c. !=
d. ^

12. Dada la siguiente expresión double x = 15/2.0:

a. x vale 7 ya que el operador / es división entera.
b. x vale 7.5 ya que al ser uno de los operandos de tipo real la división será real.
c. No se puede evaluar porque 15 es de tipo entero y 2.0 es de tipo real.
d. Todas las anteriores son falsas.

13. Los tipos de comentarios que hay son

a. Una o dos líneas (/), Tres líneas o mas (/* /) y Javadoc (// /).
b. Una sola línea (//), múltiples líneas (/* */) y Javadoc (/ */).**
c. Una o dos líneas (//), Tres líneas o mas (/* /) y Javadoc (// /).
d. Una sola línea (/), múltiples líneas (/* /) y Javadoc (// /).

14. Los tipos de datos primitivos son

a. boolean, String, byte, short, int, long, float, double.
b. **boolean, char, byte, short, int, long, float, double.**
c. boolean, char, byte, short, int, long, float, double, array, String.
d. boolean, char, byte, short, int, long, float, double, array.

15. Señala el valor de las siguientes expresiones en Java, suponiendo a y b variables de tipo booleano:
a. a=true, b=false, a || b es false.
b. a=true, b=false, a || es true.
c. a=true, b=false, a && b es true.
d. a=true, b=false, a || b es false.

16. En la definición de una clase debemos tener en cuenta que:
a. Se deben incluir los atributos comunes del conjunto de objetos y los métodos que operan sobre ellos.
b. Crearemos la clase con la palabra reservada classes.
c. El archivo de la clase debe tener el mismo nombre que el método que contenga dicha clase.
d. Todas son ciertas.

17. Indica cuál de las siguientes afirmaciones es una ventaja del ocultamiento de la información:
a. Simplifica la percepción del cliente respecto del método.
b. Permite crear una clase nueva en términos de una ya existente.
c. Evita usos inadecuados de los datos.
d. Todas las anteriores son correctas.

18. Señala cuál de los siguientes elementos no forma parte de la declaración de un método:
a. Declaración de variables locales.
b. Secuencia de instrucciones.
c. **Declaración de atributos de la clase.**
d. Declaración de parámetros.

19. Señala cuál es la correcta de las siguientes definiciones referidas a clases y objetos:
a. Toda clase es una instancia de un único objeto.
b. Un programa orientado a objetos es una colección estructurada de objetos que definen los distintos tipos de clases que van a intervenir en la resolución del problema.
c. Toda clase que forma parte del programa tiene, en un instante dado, uno o más objetos que son instancia de ella.
d. Un programa orientado a objetos está compuesto por un conjunto de objetos que son representaciones del mundo real y que interaccionan entre sí para la resolución de un problema.

20. ¿Qué tipo de estructura no lleva a cabo ningún tipo de comprobación lógica?
a. Las estructuras de selección.
b. **Las secuencias.**
c. Las estructuras de iteración.
d. Las secuencias repetitivas.

21. A un tipo de sentencia especial de decisión y una secuencia de instrucciones que pueden ser repetidas según el resultado de la evaluación de la sentencia de decisión, se le denomina...
a. Estructura de control de flujo.
b. **Estructura iterativa.**
c. Secuencia de iteraciones.
d. Estructura selectiva.

22. Si sabemos exactamente cuántas iteraciones vamos a realizar, ¿Qué tipo de bucle debemos utilizar?
a. Un bucle while con una condición robusta.
b. Un bucle for.
c. Un bucle do-while, ya que realiza al menos una entrada en el código del bucle.
d. Todas las respuestas son correctas.

23. Qué elemento puede no existir en una sentencia de selección múltiple?

a. La expresión.
b. Los case.
c. La cláusula default.
d. El break de dos o más cases.

24. Cuántas iteraciones realiza el siguiente bucle?

```
For (i=0;i<7;i++)   { System.out.println("Imprimiendo desde dentro del bucle \n");  }
```

a. 8.
b. 7.
c. 6.
d. Ninguna, la inicialización de la variable contadora es incorrecta.

25. Cuando un método utiliza una sentencia que puede generar una excepción, pero dicha excepción no es capturada y tratada por él, sino que se encarga su gestión a quién llamó al método, decimos que se ha producido delegación de excepciones. Esta delegación se realiza a través de:

a. throw.
b. **throws.**
c. throwable.
d. @throw y throws.

26. Cuál de los siguientes modificadores no es aplicable a un atributo? {

a. public.
b. extern.
c. protected.
d. private.

27. ¿Con qué modificador puede indicarse que un atributo es constante? {

a. static.
b. **final**.
c. volatile.
d. public.

28. ¿Sobre qué elementos puede aplicarse el modificador private?

a. Clases y atributos.
b. Sólo atributos.
c. Clases, atributos y métodos.
d. Atributos y métodos.

29. ¿Con qué palabra reservada se puede hacer referencia al objeto actual dentro de sus métodos?

a. element.
b. **this**.
c. me.
d. object.

30. Cuál es la palabra reservada que se utiliza para indicar la herencia en Java?

a. Java no soporta la herencia.
b. inherits.
c. extends.
d. isSubClass.

Módulo: PROGRAMACIÓN
I.E.S.: AGUADULCE **C.F.G.S. : DESARROLLO DE APLICACIONES WEB**
Fecha: 13 DE JUNIO DE 2012 **Hora de Comienzo: 15:30 h**
Examen: TEÓRICO JUNIO **Duración: 2:30 h**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

INSTRUCCIONES:

- **El EXAMEN TEÓRICO se divide en dos parciales. El PRIMER PARCIAL corresponde a las unidades 1 a 5. El SEGUNDO PARCIAL corresponde a las unidades 6 a 9.**
- **Aquellos alumnos que aprobaron el examen de FEBRERO no están obligados a realizar este PRIMER PARCIAL.**

LAS RESPUESTAS A TODAS LAS PREGUNTAS SE DARÁN EN LA TABLA DE RESPUESTAS. NO SE TENDRÁ EN CUENTA NINGUNA RESPUESTA FUERA DE ESTA TABLA.

EVALUACIÓN PRIMER PARCIAL

La puntuación total del examen TEÓRICO + PRÁCTICO será de 10 puntos. Parte teórica: 3.5 puntos**. Parte Práctica:** 6.5 puntos.

- **El examen teórico consta de 25 preguntas tipo test (2.5 puntos) más una pregunta de conocimientos prácticos (1 punto).**
- **Cada pregunta correcta del tipo test vale 0.1 puntos y cada pregunta incorrecta -0.05 puntos.**

RESPUESTAS A LAS PREGUNTAS TIPO TEST

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
d	b	a	d	b	d	b	b	a	c	c	b	b	d	a	b	b	b	c	a	b	c	b	b	d

RESPUESTA A LA PREGUNTA PRÁCTICA

26	**SOLUCIÓN: 6**

EVALUACIÓN SEGUNDO PARCIAL

La puntuación total del examen TEÓRICO + PRÁCTICO será de 10 puntos. Parte teórica: 3.5 puntos**. Parte Práctica:** 6.5 puntos.

- **El examen teórico consta de 25 preguntas tipo test (2.5 puntos) más cuatro preguntas de conocimientos prácticos (1 punto).**
- **Cada pregunta correcta del tipo test vale 0.1 puntos y cada pregunta incorrecta -0.05 puntos.**

RESPUESTAS A LAS PREGUNTAS TIPO TEST

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
b	c	b	c	b	b	a	a	c	a	a	c	b	c	c	b	b	c	b	c	d	c	c	c	b

RESPUESTAS A LAS PREGUNTAS PRÁCTICAS

26a	protected abstract double getSalario();
26b	public static final double SALARIO_MINIMO=635.0;
27a	this(nombre); this.sueldoBase = sueldo
28a	super(nombre, sueldo); this.complementoSueldo = complemento;

28b	Empleado cajero1 = new Cajero(“Pepico Pérez”, 1001.0, 300);
29a	String imprime = super.toString();

Módulo: PROGRAMACIÓN
I.E.S.: AGUADULCE **C.F.G.S. : DESARROLLO DE APLICACIONES WEB**
Fecha: 13 DE JUNIO DE 2012 **Hora de Comienzo: 15:30 h**
Examen: TEÓRICO JUNIO **Duración: 2:30 h**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

PRIMER PARCIAL

1. Un lenguaje compilado recibe también el nombre de _____ y debe ser traducido a un código que pueda entender la máquina.
   a. Lenguaje ensamblador
   b. Lenguaje interpretado
   c. Lenguaje de bajo nivel
   d. Lenguaje de alto nivel

2. ¿Qué orden debemos introducir en línea de comandos para poder obtener un archivo .class, si tuviéramos un archivo llamado programa.java?
   a. javac.exe
   b. javac programa.java
   c. java programa.java
   d. ./java programa.class

3. ¿Cuál de los siguientes entornos no es de pago?

   a. BlueJ.
   b. JBuilder.
   c. JDeveloper.
   d. IntelliJ IDEA.

4. Si la solución a un problema se hace en un tiempo mínimo y de manera óptima, decimos que esta solución es_____, por el uso correcto de los recursos del sistema.
   a. Fiable.
   b. Eficaz.
   c. Correcto.
   d. Eficiente.

5. ¿En cuál de las fases de la programación se realiza la compilación del programa?
   a. En la fase de resolución del problema.
   b. En la fase de implementación.
   c. En la fase de explotación.
   d. En la fase de mantenimiento.

6. El operador _____ se usa para la toma de decisiones.
   a. :?
   b. :
   c. ?;
   d. ?:

7. Indica los valores de x y z después de las siguientes sentencias:
   int x = 12;
   int z = ++x%7;
   a. x es 13 y z es 5.
   b. x es 13 y z es 6.
   c. x es 12 y z es 6.
   d. x es 12 y z es 5.

8. Señala cuál no es un tipo primitivo en Java
   a. short.
   b. string.
   c. double.
   d. boolean.

9. En la definición de una clase debemos tener en cuenta que:
   a. Se deben incluir los atributos comunes del conjunto de objetos y los métodos que operan sobre ellos.
   b. Crearemos la clase con la palabra reservada classes.
   c. El archivo de la clase debe tener el mismo nombre que el método que contenga dicha clase.
   d. Todas son ciertas.

10. De las siguientes afirmaciones referidas a los métodos, señala cuál es la correcta:
    a. Los atributos de instancia junto con los métodos de instancia reciben el nombre de miembros de clase.
    b. Cualquier método puede no devolver un valor, en cuyo caso se indica sin utilizar ninguna palabra reservada.
    c. La lista de parámetros de un método debe coincidir con la lista de argumentos con los que es llamado.
    d. Todas son correctas.

11. De las siguientes afirmaciones referidas a los métodos, señala cuál es la correcta:
    a. Los atributos de instancia junto con los métodos de instancia reciben el nombre de miembros de clase.
    b. Cualquier método puede no devolver un valor, en cuyo caso se indica sin utilizar ninguna palabra reservada.
    c. La lista de parámetros de un método debe coincidir con la lista de argumentos con los que es llamado.
    d. Todas son correctas.

12. Una variable local almacena un valor temporal y se declara dentro de
    a. Una clase.
    b. Un método.
    c. Un tipo de datos.
    d. Un bloque de código entre corchetes.

13. Las cadenas de caracteres se representan mediante la clase
    a. Array.
    b. String.
    c. Scanner.
    d. Math.

14. Señala cuál es la correcta de las siguientes definiciones referidas a clases y objetos:
    a. Toda clase es una instancia de un único objeto.
    b. Un programa orientado a objetos es una colección estructurada de objetos que definen los distintos tipos de clases que van a intervenir en la resolución del problema.
    c. Toda clase que forma parte del programa tiene, en un instante dado, uno o más objetos que son instancia de ella.
    d. Un programa orientado a objetos está compuesto por un conjunto de objetos que son representaciones del mundo real y que interaccionan entre sí para la resolución de un problema.

15. Indica cuál es equivalente al operador condicional de Java:
    a. if o if-else.
    b. switch.
    c. if, if-else y switch.
    d. Todas las respuestas son correctas.

16. ¿Qué es necesario incluir en cada conjunto de sentencias asociadas a los posibles valores que pueden tomarse en un switch?
    a. Puntos y comas.
    b. Una sentencia break.
    c. Una sentencia continue.
    d. Una cláusula default.

17. ¿En qué bucle se lleva a cabo la inicialización de una variable en su cabecera?

    a. En el bucles for/in.
    b. En el bucle for.
    c. En el bucle while.
    d. En el bucle do-while.

18. ¿Qué tipo de estructura no lleva a cabo ningún tipo de comprobación lógica?
    a. Las estructuras de selección.
    b. Las secuencias.
    c. Las estructuras de iteración.
    d. Las secuencias repetitivas.

19. Para declarar una nueva clase se utiliza la palabra reservada:
    a. new.
    b. object.
    c. class.
    d. classdef.

20. if (numero % 2 == 0) System.out.print(“El número es par /n”);

    a. Muestra el mensaje por pantalla cuando el número almacenado en la variable número es par.
    b. Muestra el mensaje por pantalla cuando al dividir el valor de la variable número entre 2 obtenemos cero como resultado.
    c. No muestra el mensaje por pantalla, ya que la condición del if nunca se cumplirá.
    d. Ninguna respuesta es correcta.

21. ¿Qué tipo es devuelto por un constructor?
    a. void.
    b. No devuelve ningún tipo (ni siquiera void).
    c. Depende de si el constructor está sobrecargado o no.
    d. El mismo tipo que el atributo principal de la clase.

22. Los métodos especiales que permiten la creación de un objeto y que tienen el mismo nombre que la clase a la que pertenecen son conocidos como…
    a. Iniciadores.
    b. Compiladores.
    c. Constructores.
    d. Extractores.

23. ¿Es posible utilizar una return en cualquier punto de un método?, con lo que éste finalizará en el lugar donde se encuentre dicho return. ?
    a. No, siempre debe ir al final del método.
    b. Sí y hará que éste finalice en el lugar donde se encuentre el return.
    c. Sí y podemos añadir tantos return como necesitemos.
    d. No, return sólo se incluirá en aquellos métodos que devuelven void.

24. ¿Qué modificadores habría que añadir a un método que se desea que sea accesible desde fuera de la clase y que no haga falta que existan objetos de la clase para poder ser utilizado?
    a. protected final.
    b. public static.
    c. private static.
    d. protected.

25. ¿Qué palabra reservada hay que utilizar para llamar a un constructor de una clase?
    a. build.
    b. create.
    c. free.
    d. new.

26. Dado el siguiente código en Java, indica el resultado imprimido en pantalla. (1 punto)

```
public class Sumas{
 public static void main(String[] args){
   int suma;
   for (int i=1;i<=10;i++){
     suma = 0;
     for (int j=i-1; j>=1; j--){
       if (i%j==0){
          suma=suma+j;
       }
     }
   if (suma==i){
      System.out.print(i+" ");
   }
  }
 }
}
```

SEGUNDO PARCIAL

1. Para averiguar la codificación que posee un fichero, podemos usar:
   El método Unicode().
   El método getEncoding().
   El método getCodification().
   El método getCharacter().

2. Para averiguar si un elemento es un directorio usamos el método:
   a. mkdirs.
   b. mkdir.
   c. isDirectory.
   d. isFolder.

3. Al método setLayout()
   a. Se le puede indicar si se desea que la aplicación sea portable o no.
   b. Se le pasa como argumento un objeto del tipo de Layout que se quiere establecer.
   c. Se le pasa como argumento un String para indicarle la forma del layout que se quiere establecer.
   d. Ninguna es cierta.

4. Un botón JButton y un JToggleButton
   a. Son lo mismo.
   b. Son botones, pero el primero se comporta como un interruptor de dos posiciones.
   c. Son botones, pero el segundo se comporta como un interruptor de dos posiciones.
   d. Todas son correctas.

5. Swing es
   a. Un componente de SWT.
   b. una librería de Java para la generación del GUI en aplicaciones.
   c. Una librería de NetBeans.
   d. Ninguna afirmación es correcta.

6. Si al intentar acceder a un fichero, no existe se generará una:
   a. RMIException.
   b. IOException.
   c. SQLException.
   d. FILException.

7. Las casillas de verificación en Swing están implementadas para Java por la clase:
   a. JCheckBox
   b. JScrollPanel
   c. JSeparador
   d. JButton

8. El componente Swing que dibuja una línea horizontal en el menú es:
   a. JSeparator
   b. JSeparador
   c. JDistinct
   d. JHorLine

9. La capacidad de las estructuras denominadas dinámicas...
   a. es infinita.
   b. se establece en el momento de la creación.
   c. crece conforme insertamos nuevos elementos.
   d. depende de los elementos que se inserten.

10. ¿Cuál de las siguientes expresiones encajan con la expresión regular "[A-Z0-9]+0-9"?
    a. "AZ090-9"
    b. "0AZZ-9"
    c. "AAA09"
    d. "ABC9+0"

11. A continuación, se muestra un listado de métodos que permiten comprobar, a través de la clase Matcher, si una cadena encaja con un patrón, ¿cuál de ellos debe usarse para hacer uso de los métodos start y end, también disponibles en la clase Matcher? (Imagina que m es una instancia de la clase Matcher.)
    a. m.find()
    b. m.search()
    c. m.lookingAt()
    d. m.matches()

12. Dado el array int j[]={1,2,3,4,5,6};, ¿cuál es el elemento en la posición 3?
    a. 2
    b. 3
    c. 4
    d. La inicialización del array es incorrecta.

13. ¿Cuál de las siguientes afirmaciones sobre el método toString() es falsa?
    a. Está disponible en cualquier clase de Java.
    b. Los tipos de datos primitivos, int, long, etc. pueden pasarse a cadena con este método directamente.
    c. Sirve para convertir un objeto a cadena, es especialmente útil en las clases envoltorio de los datos primitivos.
    d. Este método está disponible en la clase String.

14. ¿Cuáles de las siguientes especificaciones de formato para el método format genera un número con dos decimales?
    a. System.out.println(String.format("%.2d",2));
    b. System.out.println(String.format("%2s",2));
    c. System.out.println(String.format("%.2f",2));
    d. System.out.println(String.format("%2b",2));

15. En Programación Orientada a Objetos, ¿con qué nombre es conocido el mecanismo que permite crear clases basadas en otras existentes?
    a. Polimorfismo.
    b. Derivación.
    c. Herencia.
    d. Encapsulación.

16. ¿Para qué estructura existe la herencia múltiple en Java?
    a. Para clases.
    b. Para interfaces.
    c. Para clases que implementen la interfaz Multiple.
    d. En ningún caso.

17. ¿Qué modificadores incluyen implícitamente los métodos de una interfaz en Java y por tanto no es necesario indicarlos?
    a. protected y final.
    b. public y abstract.
    c. public y final.
    d. protected y abstract.

18. Cuando una clase está definida dentro de otra, ¿qué tipo de relación se suele decir que existe entre esas dos clases?
    a. Herencia.
    b. Derivación.
    c. Anidación.
    d. Composición.

19. ¿Con qué nombre son conocidas aquellas clases cuya única función es la de ser superclase en una jerarquía, sin que llegue a haber nunca instancias de ellas?
    a. clases básicas.
    b. clases abstractas.
    c. clases jerárquicas.
    d. Ese tipo de clases no tienen sentido y no existen en Java.

20. ¿Cuál es la palabra reservada que se utiliza para indicar la herencia múltiple de clases en Java?
    a. extendsMultiple.
    b. inherits.
    c. Java no soporta la herencia múltiple de clases.
    d. isSubClass.

21. ¿Qué palabra reservada hay que utilizar en Java para referirse a la superclase de la clase actual?
    a. superclass.
    b. that.
    c. this.
    d. super.

22. ¿Qué hay que hacer en Java para crear un objeto polimórfico?
    a. Utilizar la palabra reservada polimorphic.
    b. Declarar una variable como referencia a un objeto de una clase determinada y posteriormente asignar a esa variable referencias a objetos de otras clases diferentes.
    c. Declarar una variable como referencia a un objeto de una clase determinada que tenga clases derivadas y así posteriormente se podrán asignar a esa variable referencias a objetos de subclases de la clase referencia inicial.
    d. En Java no es posible el polimorfismo.

23. Dada la expresión regular "([A-Z]*)([a-z]*)([0-9]+)", al usar el método find() de la clase Matcher sobre la cadena "AABBccdd1234" , ¿cómo puedes extraer las letras en minúsculas? (Imagina que m es la instancia de la clase Matcher).
    a. No es posible, dado que la cadena no encaja con el patrón dado.
    b. m.group(1)
    c. m.group(2)
    d. m.get(1)

24. ¿Cuáles de los siguientes métodos nos permiten insertar elementos de un TreeSet?
    a. append()
    b. insert()
    c. add()
    d. offer()

25. ¿Cuál de las siguientes afirmaciones sobre documentos XML DOM es falsa?
    a. Tienen un único elemento raíz.
    b. Los atributos pueden estar dentro de comentarios (clase Comment) y dentro de elementos (clase Element).
    c. Puede haber elementos (clase Element) dentro de otros elementos.
    d. Un documento XML DOM es una estructura jerárquica donde todos los elementos extienden la clase Node.

26. Dado el siguiente código en Java:

```
abstract class Empleado{
        private String nombre;
        private double sueldoBase;

        public Empleado(String nombre) {
            this.nombre=nombre;
            sueldoBase=0;
        }
        public String toString(){
            String imprime = "Nombre: " + this.nombre + "Sueldo base: " + this.sueldoBase;
            return imprime;
        }
}

class Cajero extends Empleado{
        private int complementoSueldo;
}

class Interventor extends Empleado{
        private double comisionVentas;
}
```

a. **Escribe la sentencia que defina un método abstracto para la clase Empleado, que se llame getSalario, que duelve el tipo double, y que sea accesible solo por las subclases que pertenezcan al mismo paquete que la clase Empleado.**

b. **Escribe la sentencia que defina una constante de clase pública llamada SALARIO_MINIMO, de tipo double inicializada a 635.0.**

27. Dada la siguiente definición de la cabecera de otro constructor para la clase Empleado:

```
public Empleado(String nombre, double sueldo) {
        //cuerpo del constructor
}
```

a. **Escribe las sentencias necesarias en el cuerpo para completar la definición, utilizando para ello una llamada al primer constructor de la clase.**

28. Dada la siguiente definición de la cabecera del constructor para la clase Cajero:

```
public Cajero(String nombre, double sueldo, int complemento) {
        //cuerpo del constructor
}
```

a. **Escribe las sentencias necesarias en el cuerpo para completar la definición, utilizando para ello una llamada al segundo constructor de la clase padre.**

b. **Utiliza el constructor que acabas de completar, para crear un objeto de tipo *Cajero* llamado cajero*1* cuyo nombre es *Pepico Pérez,* cuyo sueldo base *1001.0* y cuyo complemento es *300*. El objeto creado será referenciado por una variable de tipo Empleado.**

29. Necesitamos que el método toString de la clase Cajero heredado de la clase Empleado devuelva también el valor del atributo complementoSueldo dentro de la cadena de caracteres. Dada la siguiente definición de la cabecera y el cuerpo del método:

```
public String toString(){
        // Sentencia que falta
        imprime = imprime + "Complemento del sueldo: " + this.complementoSueldo;
        return imprime;
}
```

a. **Escribe las sentencia necesaria que falta en el cuerpo del método que declare la variable imprime de tipo String y a la que se le asigne la cadena de caracteres devuelta en la llamada al método toString de la clase Empleado.**