- Cada **ejercicio** incluye una **leyenda** con: el *resultado de aprendizaje* que se evalúa, el *tiempo recomendado para la realización* del ejercicio y su *puntuación*.
- Para superar el examen **deberá obtener, al menos, la mitad de los puntos propuestos para cada resultado de aprendizaje**.

**Resultados de aprendizaje** evaluados en este examen (test/práctica):

*R1* Ser capaz de escribir programas de ordenador que funcionen correctamente en un ordenador.

*R3* Comprender el paradigma de programación orientado a objetos, incluyendo conceptos como herencia y polimorfismo.

*R4* Utilizar un sistema de archivos para el almacenamiento persistente de información.

*R5* Saber implementar una solución a un problema de ingeniería empleando un lenguaje de programación orientado a objetos.

*R6* Saber emplear en la resolución de problemas una librería de estructuras de datos avanzada de un lenguaje de programación.

*R7* Saber ordenar un conjunto de datos y saber buscar un dato de modo eficiente.

1. *R1* Sea el siguiente código:

```java
public class Principal{
    public static int m(int a, int b){
        if (b == 0) {
            return 1;
        }
        int acumulador = 1;
        for(int i = 1; i <= b; i++) {
            acumulador = acumulador * a;
        }
        return acumulador;
    }
    public static void main(String[] args){
        System.out.println(10*m(12,2));
    }
}
```

(a) Se imprime 12120 por pantalla.

(b) Se imprime 1440 por pantalla.

(c) Se imprime 21210 por pantalla.

(d) El resto de las respuestas son falsas.

2. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```java
public class Objeto1 {

    public static int attribute1;

    public Objeto1(int attribute1){
        Objeto1.attribute1 = attribute1;
    }

    public static void main(String[] args) {
        Objeto1 obj1 = new Objeto1(2);
        Objeto1 obj2 = new Objeto1(1);
        System.out.println(Objeto1.attribute1);
    }
}
```

   (a) 1

   (b) 2

   (c) No compilará porque no puede accederse a variables públicas de la clase.

   (d) No compilará porque no se ha establecido el atributo final.

3. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```java
package org.example;

public class Objeto2 {

    public int attribute1;

    public Objeto2(int attribute1) {
        this.attribute1 = attribute1;
    }

    public static int getAttribute1()
    {
        return this.attribute1;
    }

    public static void main(String[] args) {
        Objeto2 obj1 = new Objeto2(1);
        Objeto2 obj2 = new Objeto2(2);
        System.out.println(obj1.getAttribute1());
    }
}
```

   (a) 1

   (b) 2

   (c) No compilará porque no puede accederse a atributos desde funciones estáticas.

   (d) No compilará porque no puede accederse a variables públicas de las clases.

4. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```
1  package org.example;
2
3  public class V3D {
4
5      public static void main(String[] args){
6          V2D v1 = new V2D(1,1);
7          V3D v2 = new V3D(2,2,2);
8          V3D vResult = v2.resta(v1);
9          System.out.println(vResult);
10     }
11     protected int x;
12     protected int y;
13     protected int z;
14
15     public V3D(int x, int y, int z)
16     {
17         this.x = x;
18         this.y = y;
19         this.z = z;
20     }
21
22     public V3D suma(V3D v1){
23         return new V3D(this.x+v1.x,this.y+v1.y,this.z+v1.z);
24     }
25
26     @Override
27     public String toString() {
28         return x + ", " + y + ", " + z;
29     }
30 }
31
32 class V2D extends V3D {
33
34     public V2D(int x, int y){
35         super(x,y,0);
36     }
37
38
39     public V3D resta(V3D v1){
40         return new V3D(this.x - v1.x, this.y - v1.y, - v1.z);
41     }
42
43 }
```

(a) Error de compilación porque v3 no define el método resta.

(b) 1,1,2

(c) Error de compilación porque v2 debe devolver un objeto de tipo V2D.

(d) Error de compilación porque v2 recibe un V3D en la definición de su método suma y está recibiendo un objeto de tipo V2D.

5. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```
1  package org.example;
2
3  public class Empleado {
4
5
6      public static void main(String[] args) {
7          Jefe e1 = new Jefe(1000,3);
8          System.out.println(e1.getSueldo());
9      }
10
11     private int sueldoBase;
12
13     public Empleado(int sueldo) {
14         this.sueldoBase = sueldo;
15     }
16
17     public int getSueldo()
18     {
19         return this.sueldoBase;
20     }
21
22 }
23
24
25 class Jefe extends Empleado {
26     private int years;
27     public Jefe(int sueldo, int years) {
28         super(sueldo);
29         this.years = years;
30     }
31
32     public int getSueldo(int moneyPerYear)
33     {
34         return this.getSueldo() + this.years*moneyPerYear;
35     }
36
37 }
```

(a) 1000

(b) 1300

(c) No compila porque la clase Empleado no tiene un atributo con nombre **sueldo**.

(d) No compila porque el método getSueldo() invocado no está definido en la clase Jefe.

6. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```java
package org.example;

public class Empleado {

    public static void main(String[] args) {
        Empleado e1 = new Empleado(100);
        System.out.println(e1.getSueldo(100));
    }

    private int sueldoBase;

    public Empleado(int sueldo) {
        this.sueldoBase = sueldo;
    }

    public int getSueldo()
    {
        return this.sueldoBase;
    }
}


class Jefe extends Empleado {
    private int years;
    public Jefe(int sueldo, int years) {
        super(sueldo);
        this.years = years;
    }

    public int getSueldo(int moneyPerYear)
    {
        return this.getSueldo() + this.years*moneyPerYear;
    }

}
```

(a) 1000

(b) 1300

(c) No compila porque la clase Empleado no tiene un atributo con nombre **sueldo**.

(d) No compila porque el método getSueldo(int ...) invocado no está definido en la clase Empleado.

7. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```java
package org.example;

import java.util.Arrays;

public class Exceptions {


    public static boolean is_multiple(int n, int m) throws
            IllegalArgumentException {
        if (n < 0 || m < 0) {
            throw new IllegalArgumentException("n or m cannot be negative");
        } else if (m % n == 0) {
            return true;
        } else {
            return false;
        }
    }

    public static void main(String[] args) {
        try {
            boolean is_multiple = is_multiple(-10,-5);
            System.out.println(is_multiple);
        }
        catch(IllegalArgumentException iae){
            System.out.println(iae.getMessage());
        }
    }
}
```

(a) -5,-2

(b) true

(c) n or m cannot be negative.

(d) NullPointerException...

8. *R1, R3, R5* Dado el siguiente código, indicar qué aparecerá en la consola:

```java
package org.example;

import java.util.Arrays;

public class Exceptions {


    public static boolean is_multiple(int n, int m) throws
            IllegalArgumentException {
        if (n < 0 || m < 0) {
            throw new IllegalArgumentException("n or m cannot be negative");
        } else if (m % n == 0) {
            return true;
        } else {
            return false;
        }
    }

    public static int[] divisors(int m) throws IllegalArgumentException {
        int[] divisors = new int[1000];
        int indexDivisors = 0;
        for (int i = 2; i < m; i++) {
            boolean is_i_multiple = is_multiple(i, m);
            if (is_i_multiple) {
                divisors[indexDivisors] = i;
                indexDivisors++;
            }
        }
        // make a copy of an array with the actual dimension
        int[] actualDivisors = new int[indexDivisors];
        for (int auxIndice = 0; auxIndice < indexDivisors;
             auxIndice++) {
            actualDivisors[auxIndice] = divisors[auxIndice];
        }
        return actualDivisors;
    }

    public static void main(String[] args) {
        try {
            int[] divisors = divisors(-10);
            System.out.println(Arrays.toString(divisors));
        }
        catch(IllegalArgumentException iae){
            System.out.println("");
        }
    }

}
```

(a) -5,-2

(b) [ ]

(c) n or m cannot be negative.

(d) NullPointerException...

9. *R1* Dado el siguiente código, especifique cuándo se produce el caso general:

```java
public class Recursividad {

    public static int foo(int n){
        if(n < 0){
            return 0;
        }
        else if(n == 0)
        {
            return foo(n-2) / 2;
        }
        else{
            return 1 + foo(n-1);
        }
    }
}
```

(a) n = 0

(b) n > 0

(c) n < 0

(d) n >= 0

10. *R1* Dado el siguiente código, especifique el orden de complejidad computacional del algoritmo:

```
1  package org.example;
2
3  public class ComplComp {
4
5      public void foo(int whatever){
6          int fooVariable = 1;
7          for(int index1=1; index1<whatever;index1+=2){
8              for(int index2=0;index2<index1;index2++){
9                  fooVariable +=index2*index1;
10             }
11         }
12     }
13 }
```

    (a) O(n*log(n))

    (b) O($n^2$)

    (c) O(n)

    (d) O(1)

11. *R1* Dados dos algoritmos A y B con las operaciones:
    A = $2n^2$
    B = 16*log(n)
    Indique A partir de qué número es más eficiente B que A para valores de n > 1.

    (a) 1

    (b) 2

    (c) 4

    (d) 6

12. *R7* Dado el siguiente conjunto de elementos, indique cuál sería el algoritmo de búsqueda más eficiente de entre las opciones indicadas:
    1 3 6 20 25 30 31 40

    (a) Quicksort

    (b) Búsqueda binaria siempre que la estructura tenga acceso aleatorio

    (c) Secuencial sin centinela

    (d) Secuencial con centinela

13. *R7* Dado el siguiente conjunto de elementos y el siguiente algoritmo de inserción, indique cuál será el orden de complejidad computacional para el caso concreto:

```
1  private static void insercion(int[] datos) {
2    int i, temp;
3    for (int elementoAInsertar = 1; elementoAInsertar < TAM; elementoAInsertar++) {
4      temp = datos[elementoAInsertar];
5      int dondeInsertar = 0;
6      while (datos[dondeInsertar] < temp) {
7        dondeInsertar++;
8      }
9      if (dondeInsertar < elementoAInsertar) {
10       for (i = elementoAInsertar; i >
11         dondeInsertar; i--) {
12         datos[i] = datos[i - 1];
13       }
14     datos[dondeInsertar] = temp;
15     }
16   }
17 }
```

8 7 6 5 4 3 2 1

(a) O(n)

(b) O($n^2$)

(c) O($n^2/2$)

(d) O(log(n))

14. *R5, R6* Sea el siguiente código
List lista = new ArrayList<Integer>();
donde 'ArrayList' y 'List' hacen referencia a 'ArrayList.java'y 'List.java'de la API de Java respectivamente.

(a) No es correcto porque no se puede crear ningún objeto que sea instancia directa de 'List.java'.

(b) Crea un arrayList.

(c) Crea una lista enlazada.

(d) El resto de las respuestas son falsas.

15. *R6* En la clase 'ArrayList.java'

(a) Se aplica polimorfismo con respecto a 'List.java', pero no polimorfismo paramétrico.

(b) No se aplica ningún tipo de polimorfismo.

(c) Se aplica tanto polimorfismo de herencia con respecto a 'List.java' como polimorfismo paramétrico.

(d) Todas las respuestas son falsas.

16. *R5, R6* Con respecto a 'TreeMap.java' y 'TreeSet.java', y asumiendo que en el mapa la cuestión utiliza la repetición como repetición de claves se puede afirmar que:

(a) En ninguno de los dos casos se mantienen elementos repetidos.

(b) En 'TreeMap.java' se mantienen elementos repetidos, pero en 'TreeSet.java' no.

(c) En 'TreeSet.java' se mantienen elementos repetidos, pero en 'TreeMap.java' no.

(d) En ambas estructuras se mantienen elementos repetidos.

17. *R1, R3, R5* Especifique si la clase aquí definida como V2D es mutable:
    (a) Sí porque no es una clase final y otras clases pueden heredar de ella.
    (b) Sí porque se pueden añadir nuevos métodos.
    (c) No porque no es una clase abstracta.
    (d) No porque sus atributos son finales.

18. *R1, R3, R5* Especifique la identidad de V2D:
    (a) X,Y
    (b) X
    (c) Y
    (d) Referencia

19. *R1, R3, R5* Asumiendo que se crea la tabla hash especifica en el programa principal main con un factor de carga 0,75 y un tamaño 4, ¿En qué posición de la tabla se encontrada el objeto v1?
    (a) 1
    (b) 2
    (c) 3
    (d) 7

20. *R1, R3, R5, R7* Asumiendo que se crea la tabla hash especifica en el programa principal main con un factor de carga 0,75 y un tamaño 4, ¿En qué posición de la tabla se encontrada el objeto v2?
    (a) 1
    (b) 2
    (c) 3
    (d) 7

21.

1. (2 puntos) + 1 punto extra (*R1, R4, R6*, 30 min) Se solicita una función recursiva que reciba una cadena de caracteres y devuelva la cadena en el orden inverso, escogiendo el tipo de datos que mejor se ajuste según las necesidades. Justifique sus respuestas de manera concisa y concreta.

   Por ejemplo, la cadena "Hola" deberá devolver una estructura de datos conteniendo los caracteres "aloH". Asuma que la entrada por teclado contiene funciones getString(), getInt() y sucesivas ya implementadas.

   **No es necesario que cierre los recursos ni manejar ninguna excepción en esta cuestión**

   **Tenga en cuenta que los apartados se evaluarán de manera independiente. El estudiante puede asumir que las acciones previas han sido correctamente implementadas.**

   (a) 0.5 puntos extra. Preserve el principio de encapsulamiento. Para ello, la función recursiva debe ser visible desde el exterior para ser invocada con los parámetros mínimos necesarios.

   No es necesario rellenar esta respuesta. Aplica sobre la solución planteada.

   (b) 0.5 puntos. Establezca el caso base y el caso general, incluyendo la signatura pública de la función.

(c) 0.5 puntos. Implemente la solución en Java.

(d) 0.5 puntos. Cree una función que lea todas las líneas de un fichero cuyo nombre se pasa por parámetro, invierta las cadenas y devuelva la estructura de datos adecuada con las cadenas invertidas. Asuma que la función recursiva está correctamente implementada. En este apartado no se tendrá en cuenta la solución de los apartados anteriores.

(e) 0.5 puntos. Cree una función que escriba una cadena o un conjunto de cadenas en un fichero cuyo nombre se pase por parámetro. Indicar al final de la solución cómo se escribirá el contenido con el ejemplo de un fichero con dos cadenas y las cadenas Hola y AED invertidas. Hay libre elección de cómo escribir el contenido siempre que sea consistente con la explicación. Asuma que la función recursiva está correctamente implementada. En este apartado no se tendrá en cuenta la solución de los apartados anteriores.

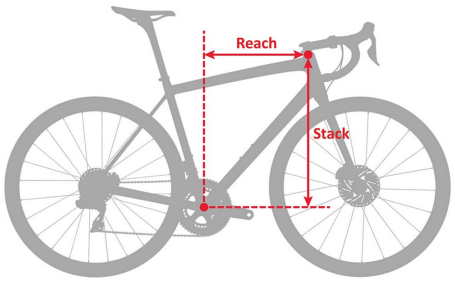
(f) 0.5 puntos extra. Cree una función main que lea el nombre del fichero origen, invoque a la función de lectura y separación de las cadenas, lea el nombre del fichero destino e invoque a la función de escritura con las cadenas de caracteres separadas. Asuma que todas las funciones anteriores están correctamente implementadas. En este apartado no se tendrá en cuenta la solución de los apartados anteriores.

**Asuma que la lectura de teclado está implementada mediante la función leerString().**

2. (2 puntos) + 1 punto extra (*R1, R3, R5*, 30 min.) Los *estudios biométricos* para ciclistas son cada vez más frecuentes para preservar la salud de los ciclistas populares. La morfología de cada cuerpo humano es diferente y la geometría de la bicicleta es clave para poder aumentar el rendimiento y, sobre todo, evitar lesiones por una mala salud postural.

   En esta aplicación, cada *Ciclista* está representado mediante su *nombre completo*, su altura en milimetros, su *actitud postural en la bicicleta* (*endurance, aggresive y extreme*), y su *Id del carnet ciclista (entero)*. Además, todos los pacientes tienen asociados una colección de *Bicicletas* (clase *Bike*).

   Las bicicletas están identificadas por un número de serie, una marca, un modelo, la talla (S,M, L o XL) la distancia horizontal entre el eje del pedalier y la potencia, llamado **Reach** y la distancia vertical entre el eje del pedalier y la potencia, llamado **Stack**. Estos valores se guardan en milímetros.




   En base a estas especificaciones se solicita que:

   - Si necesita programar algún enumerado, hágalo.
   - No gestione eventos no esperados que no estén especificados.
   - Establezca el tipo de dato adecuado para cada atributo.
   - **1 punto**. Programe la clase *Bike*:
     - Incluya todos los atributos necesarios y establezca un tipo y una visibilidad adecuada.
     - Programe el constructor de la clase para que sus valores siempre sean consistentes. Justifique sus decisiones.
     - Asuma que todos los getters están creados.
     - Programe las excepciones que crea convenientes para asegurar la consistencia de los datos de la clase *Bike*. Si se crean varias excepciones, sólo es necesario implementar una de ellas, aunque deban lanzarse todas.
     - Implemente el método `equals` para la clase *Bike*. Justifique las decisiones tomadas.
     - Implemente un comparador para comparar **Bikes** por el **Reach**. Tenga en cuenta las decisiones tomadas previamente y justifique sus acciones. Elija la opción que considere conveniente y que debe ser consistente en el ejercicio posterior.
   - **1 punto**. Programe la clase *Ciclista*:
     - Suponga que la clase *Bike* está correctamente implementada.

- **U**tilice las colecciones de datos que crea más adecuadas. Una persona no puede tener asociada más de una vez la misma bicicleta. Es crucial la eficiencia a la hora de acceder a los diferentes dispositivos y el número de dispositivos no se espera que sea alto (<10).
- Incluya todos los atributos necesarios y establezca un tipo y una visibilidad adecuada.
- Asuma que existe un constructor que construye el objeto de manera consistente y con la colección de bicicletas vacía.
- Asuma que existen todos los getters y setters necesarios y que los setters garantizan la consistencia del objeto.
- Escriba un método (`numBikes`) que devuelva el número de bicicletas que tiene asignadas un ciclista.
- Escriba el método `addBike` que permite añadir una bicicleta al paciente. Esta bicleta no puede ser repetida.
- **0.5 puntos extra**. Implemente la el método `public static boolean fitsBike()` que devolverá si la bicicleta al perfil del ciclista. Este método deberá
  - Comprobar su actitud postural en la bicicleta.
  - Si la actitud postural es endurance: la proporción **stack/reach** debe estar entre 1,50 y 1,7.
  - Si la actitud postural es aggresive: la proporción **stack/reach** debe estar entre 1,40 y 1,50.
  - Si la actitud postural es extreme: la proporción **stack/reach** debe estar entre 1,30 y 1,40.
  - Además, este método comprobará la talla del ciclista. Si el ciclista mide <= 170 mm la talla ideal será la S; (170-178) será M; (178-185) será L; y > 185 será XL.
- **0.5 puntos extra**. Escriba el método (`Collection<Bike> findBikeFitting(Collection<B bikesToTest)`) que, dada una colección de bicicletas, el método devuelva la colección de bicicletas que encajan con las características del ciclista que invoca dicho método. Utilice las estructuras de datos que considere convenientes y justifique su respuesta.

A lo largo de este ejercicio, **asegúrese de que sus clases respetan el principio de encapsulamiento**. Por otra parte, **el código deberá estar correctamente estructurado en métodos**. Si no se implementa un método correspondiente, no afectará a la evaluación del resto de soluciones que llamen a ese método si su llamada conceptual es correcta.