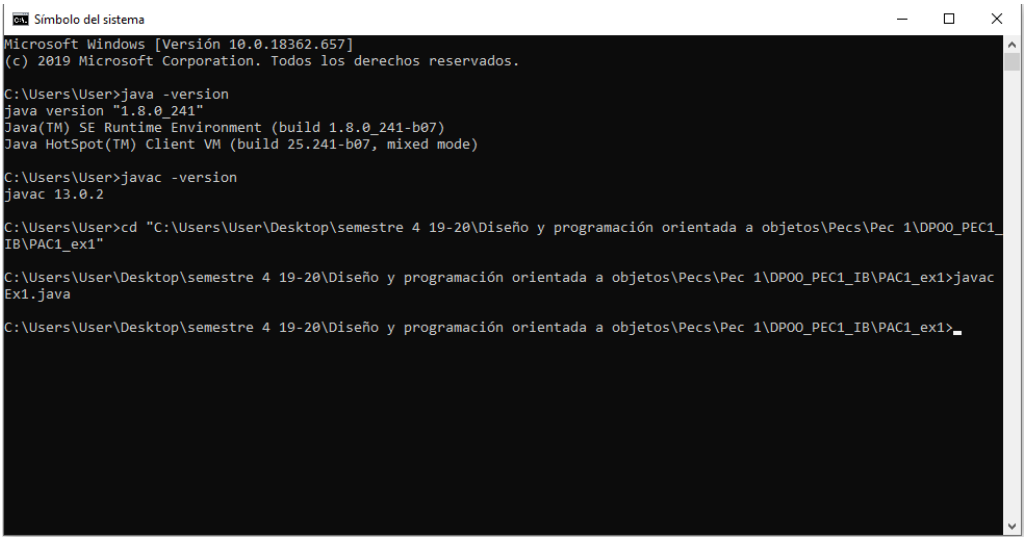
Nombre: Álvaro Giménez Gorrís

Login: agimenezgor

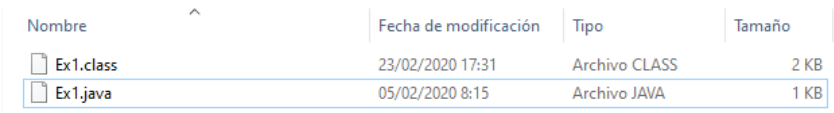
Ejercicio 1 – Instalando Java (1 punto)

a) Capturas de pantalla que demuestren que has instalado correctamente el entorno de programación Java.

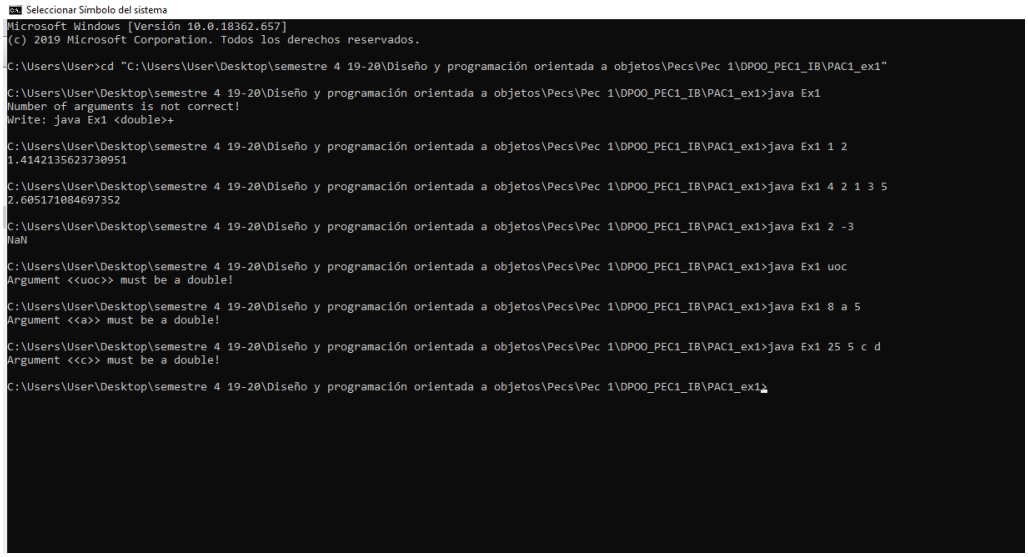
(1) Captura de pantalla en la que se vea que se ha compilado correctamente el programa vía línea de comandos.



(2) Captura de pantalla en la que se vea la carpeta con el fichero `Ex1.java` junto al fichero `Ex1.class`.



b) Captura de pantalla donde se vea la consola de línea de comandos con diferentes ejecuciones del programa `Ex1` (parecido al ejemplo que te damos en formato texto en el enunciado):



c) Explica qué hace el programa `Ex1.java` (cuanto más detalle des, mejor).

El programa consta de una clase llamada Ex1 que contiene la función **main** que recibe como entrada en vector de cadena de caracteres llamado **args**.

En primer lugar, se declara una variable de coma flotante llamada **result** al que se le asigna el valor 0 en la que se almacenará el valor final de la función.

A continuación, se declaran dos enteros: el primero llamado i al que se le asigna el valor 0 y el segundo llamado **length** al que se le asigna el valor de la longitud del vector de entrada llamado **args**.

Después, se declara la cadena de caracteres **str** con la clase **stringBuilder** que sirve para concatenar cadenas de caracteres.

Una vez declaradas todas las variables, la se usa un condicional que evalúa si el tamaño del vector es igual a 0 mediante la variable **length**. En caso de que la condición devuelva **true**, se ejecutará el código del condicional **if** y el programa devolverá un mensaje de error en el que se indica que el número de argumentos introducidos no es correcto y se indican los argumentos que se deberían de haber introducido. En caso de que la condición devuelva **false**, se ejecutará el código de la condición **else** y la función hará lo siguiente:

En primer lugar, se evalúan las excepciones mediante la estructura **try** y **catch**. El programa intentará ejecutar el código que hay dentro del método **try** y si no lo consigue ejecutará el código del método **catch.**

Dentro del método try el programa, el programa le asigna a la variable **result** el valor de numérico de la primero posición del vector **args** usando el método **Double.parseDouble** para convertir la cadena de caracteres en un valor de coma flotante. En caso de que el programa no consiga hacer la conversión de cadena de caracteres a coma flotante (porque el valor introducido no se puede representar como una variable de coma flotante), el programa ejecutara el código del método catch. A continuación, se usa un bloque iterativo **for** para recorrer el vector de entrada **args** desde la segunda posición del vector hasta la última. Dentro del bloque **for**, se asigna a la variable **result** el resultado de la multiplicación del valor actual de la variable por el valor de la conversión de cadena de caracteres (de cada una de las posiciones del vector **args** desde la segunda posición hasta la última) a valor de coma flotante. En caso de que no se pueda realizar la conversión, el programa ejecutará el código del método catch. Una vez se le asignado el valor de la multiplicación de todos los valores del vector de entrada, se le asigna a la variable **result** el valor de la potencia del valor de la multiplicación de todos los valores de entrada elevado a 1 dividido entre la longitud del vector. Finalmente, se concatenará el valor de la variable **result** seguido de un salto de línea dentro de la variable **str.**

Como he explicado anteriormente, en caso de que el código del método **try** no se pueda ejecutar, se ejecutará el código del método **catch**. Este método se limitará a imprimir por pantalla un mensaje de error en el que se indique la posición del vector que no se ha podido convertir de cadena de caracteres a coma flotante y se indicará que se debe de introducir como valor de coma flotante y, a continuación, se cerrará el programa.

Finalmente, se imprimirá fuera del condicional el valor concatenado de la cadena de caracteres **str** que contiene el resultado del cálculo realizado dentro del método **try**.

Ejercicio 2 – Aprendiendo Java (3 puntos)

Para responder este ejercicio **tienes que incluir en el fichero `loginUOC_PEC1.zip` una carpeta llamada `PAC1_ex2` que contenga**:

- El fichero `Mario.java` completado con comentarios adentro que expliquen qué haces en los pasos más significativos de tu solución.

- El fichero `Mario.class` que hayas obtenido después de compilar el fichero `Mario.java`.

Además, **en este documento tienes que hacer una breve explicación de los métodos que se te han pedido implementar**. Para hacer la explicación más sencilla, pon el código fuente del método:

Nota: Si has hecho una implementación parcial de algún método o no lo has sabido hacer, coméntalo.

a) Explicación y código fuente de los métodos.

En primer lugar, paso a explicar el método **isValidDirection**:

Primero, convierto la cadena de caracteres en minúscula para que, sea cual sea la entrada de datos, el texto sea válido.

A continuación, comparo el valor de la cadena de caracteres con el valor de las constantes LEFT y RIGHT. Si una de las dos constantes es igual a la cadena de caracteres de entrada, la función devolverá true y si no es igual a ninguna de las dos, la función devolverá false. Código fuente:

```
public static boolean isValidDirection(String direction) {
        // Convierto la cadena de caracteres en minuscula
        direction = direction.toLowerCase();

        /* Comparo el valor de la cadena con los enumerados de dirección*/
        if(direction.equals(LEFT) || direction.equals(RIGHT)){
            // Si el valor es correcto, devulve true
            return true;
        }
        else{
            // Si no es correcto, devuelve false
            return false;
        }
    }
```

En segundo lugar, paso a explicar la función **upstairRight:**

Primero, declaro la cadena de caracteres que imprimirá la escalera y le añado el último escalón de la escalera.

Después, declaro las dos variables auxiliares para los ciclos **for** necesarios y declaro una variable que guarde cuantas posiciones debe tener cada escalón y lo igualo a uno.

Después, creo un cicló **for** que recorrerá desde la posición uno hasta la última posición marcada por el valor de entrada de la variable **length**. Dentro de ese ciclo **for**, se abre otro ciclo **for** en el que se recorre desde la posición cero hasta la posición que deba de tener el escalón en el que nos encontramos para que se concatenen los asteriscos necesarios para formar el escalón. Una vez recorrido el ciclo **for** y con el escalón ya formado, se concatena un salto de línea, se suma una unidad a la variable position para que el siguiente escalón tenga un asterisco más y se continúa con el primer ciclo **for** hasta formar la escalera.

Finalmente, la función devuelve la cadena de caracteres que contiene la escalera. Código fuente:

```
public static String upstairRight(int length) {
        // Declaro la cadena de caracteres que imprimirá la escalera
        StringBuilder right = new StringBuilder();
        right.append("*" + "\n");

        /* Declaro las variables necesarias para las iteraciones de los ciclos for.
        Declaro una variable que guarde cuantas posiciones debe de tener el escalón.*/
        int i, j, position = 1;

        // Itero tantas veces como tamaño tenga el valor de entrada
        for(i = 1; i < length; i++){
            /* Itero tantas veces como tamaño necesite tener cada escalón*/
            for(j = 0; j <= position; j++){
                // Concateno los asteriscos necesarios
                right.append("* ");
            }
            // Concateno salto de línea
            right.append("\n");

            // Sumo una posición al escalón
```

```
                position++;
            }

            // Devuelvo la escalera
            return right.toString();
    }
```

En tercer lugar, paso a describir el funcionamiento de la función **upstairLeft**:

En primer lugar, declaro una cadena de caracteres que guardará el contenido de la escalera.

Después, declaro las cuatro variables necesarias para recorrer los cuatro ciclos **for** declarados, una variable que guarda la cantidad de espacios que tiene el escalón al inicio (al que le doy el valor de la longitud de la escalera menos uno) y una variable que guarda la cantidad de asteriscos que contiene el escalón (al que le asigno 2 como valor, ya que el primer escalón lo voy a declarar previamente).

A continuación, declaro un ciclo **for** que recorre desde cero hasta la cantidad de espacios que debe de contener el primer escalón al inicio y lo guardo en la cadena de caracteres. Después, le concateno el primer asterisco y un salto de línea para que el primer escalón quede formado.

Acto seguido, declaro un ciclo **for** para recorrer desde la posición uno hasta la longitud de la escalera. Dentro de este ciclo **for**, le resto una unidad a la variable que cuenta los espacios que van en cada escalón y le suma una unidad a la variable que cuanta la cantidad de asteriscos que debe de contener cada escalón. Después, declaro el tercer ciclo **for** que recorre desde la posición cero hasta la cantidad de espacios que deba contener el inicio del escalón con la intención de que en cada ciclo se concatene un espacio a la cadena de caracteres. Una vez guardados los espacios necesarios, declaro el cuarto ciclo **for** que recorre desde la posición uno (ya que el primer escalón ya está creado) hasta la cantidad de asteriscos que deba contener el escalón para poder guardar en la cadena de caracteres los asteriscos necesarios. Un a vez finalizado el escalón, se le concatena un salto de línea a la cadena de caracteres y se pasa a la siguiente iteración del ciclo hasta completar toda la escalera.

Finalmente, la función devuelve la cadena de caracteres que contiene la escalera.

Código fuente:

```
public static String upstairLeft(int length) {
        // Declaro la cadena de caracteres que imprimirá la escalera
          StringBuilder left = new StringBuilder();

        /* Declaro las variables necesarias para las iteraciones de los ciclos for.
        Declaro una variable que guarde cuantas espacios debe tener al principio*/
          int i, j, k, e, spaces, asterisks;
          spaces = length - 1;
          asterisks = 2;

          // Concateno los espacios de la primera iteración
          for(e = 0; e < spaces; e++){
                  left.append("  ");
          }

          // Contateno el primer asterisco
          left.append(" *\n");

          // Itero tantas veces como tamaño tenga el valor de entrada
          for(i = 1; i < length; i++){
                  /* Resto una posición a la cantidad de espacios a imprimir*/
                  spaces--;

                 /* Sumo una posición a la cantidad de asteriscos a imprimir*/
                  asterisks++;

                  // Concateno los espacios iniciales
                  for(j = 0; j < spaces; j++){
                          left.append("  ");
                  }

                  // Concateno los asteriscos necesarios
                  for(k = 1; k < asterisks; k++){
                          left.append(" *");
                  }
```

```
            // Concateno salto de linea
            left.append("\n");
        }

        // Devuelvo la escalera
        return left.toString();
    }
```

Por último, paso a describir la función **upstair**:

En primer lugar, declaro una cadena de caracteres que contendrá la escalera.

A continuación, convierto la variable que guarda la dirección de la escalera a minúscula.

Después, declaro un bloque condicional que if/else que, compara la dirección de entrada con la variable RIGHT. En caso de ser iguales, se le asigna a la cadena de caracteres que guarda la escalera el valor de la función **upstairRight.** En caso de que el valor de entrada no sea igual a la constante RIGHT, se le asigna a la cadena de caracteres que guarda la escalera el valor de la función **upstairLeft.**

Finalmente, la función devuelve la cadena de caracteres que contiene la escalera.

Código fuente:

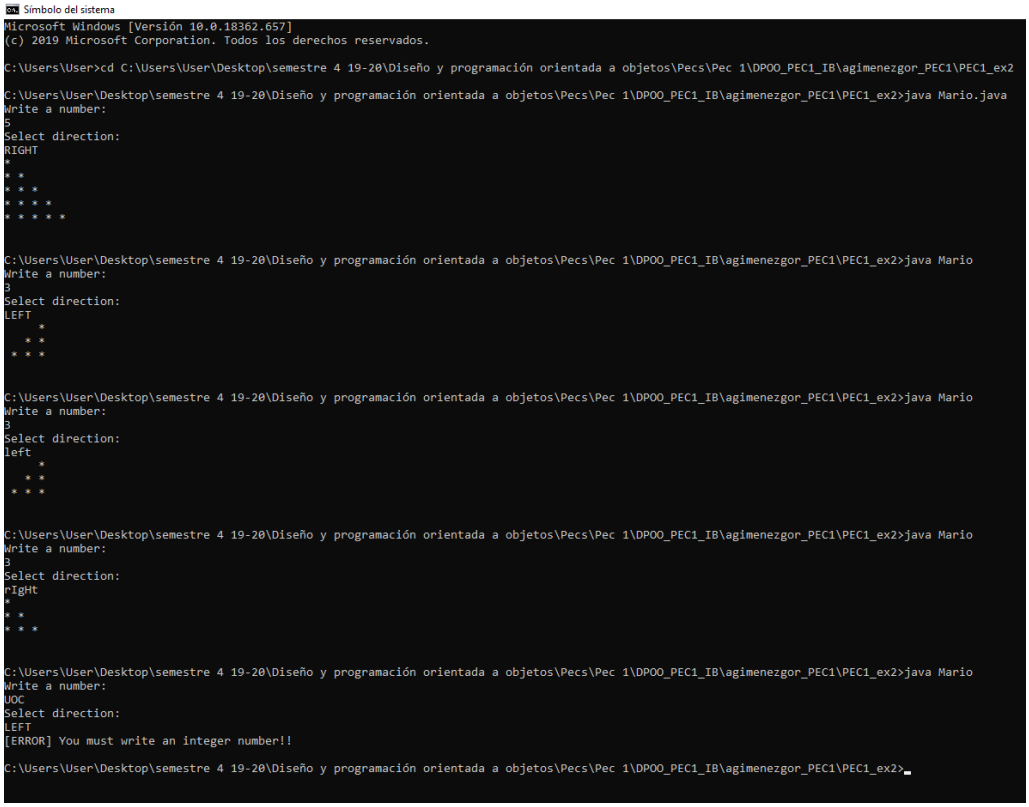
```
public static String upstair(int length, String direction) {
            // Cadena de caracteres que imprime la escalera
            String stairs;

            // Paso el valor de la cadena de caracteres a minúscula
            direction = direction.toLowerCase();

            /* Si la escalera tiene dirección derecha, llamo a la función upstairRight*/
            if(direction.equals(RIGHT)){
                stairs = upstairRight(length);
            }
            /* Si la escalera tiene dirección izquierda, llamo a la función upstairLeft*/
            else {
                stairs = upstairLeft(length);
            }
```

```
// Devuelve la escalera
return stairs;
```

b) Pon una o varias capturas de pantalla donde se vea la ejecución de tu programa.



Ejercicio 4 – Teoría sobre Diseño de Programación Orientada a Objetos (4 puntos)

a) Indica los objetos/instancias que aparecen, sólo el nombre.

Los objetos que aparecen son los siguientes:

- ***Lionel Cristiano***
- ***Sergio Abidal***
- ***Carlos Pérez***
- ***Luis González***

b) Indica las clases que ves, sólo el nombre.

Las clases que aparecen en el texto son las siguientes:

- ***Persona***
- **Jugador**
- **Colaborador**

c) Por cada una de las clases que has mencionado en el apartado B, indica sus atributos y métodos según lo que dice el texto, ¡no inventes!

Hay que tener en cuenta que el texto solo refleja los atributos de las clases y no refleja ningún método. Por lo tanto, los atributos de las clases son los siguientes:

Persona:

- ***Nombre***
- ***Salario***

Jugador

- ***Posición***
- ***Fecha de nacimiento***
- ***Fecha de inicio de representación***
- ***Transferible***

Colaborador

- ***Puesto de trabajo***

d) Para cada objeto identificado en el apartado A, indica a qué clase de las mencionadas en el apartado B pertenece.

Hay que tener en cuenta que la clase Persona es una clase abstracta y, por lo tanto, no puede ser instanciada. Además, las clases jugador y colaborador heredan los atributos de la clase Persona.

Los objetos quedan descritos a que clase pertenecen a continuación:

Lionel Cristiano:

- ***Pertenece a la clase jugador.***

Sergio Abidal

- ***Pertenece a la clase jugador.***

Carlos Pérez

- ***Pertenece a la clase colaborador.***

Luis González

- ***Pertenece a la clase colaborador.***

e) Para cada objeto indica el valor que tienen los atributos de la clase a la que pertenece. Si para un atributo no se indica el valor al texto, pone "***Unknown"***.

Atributo	Jugador
Nombre	Lionel Cristiano
Salario	10.000.000,87€
Posición	Delantero centro
Fecha de nacimiento	Unknown
Fecha inicio representación	2 de noviembre de 2002
Transferible	Si

Atributo	Jugador
Nombre	Sergio Abidal
Salario	1.000.000€
Posición	Unknown
Fecha de nacimiento	20 de noviembre de 2004
Fecha inicio representación	5 de enero de 2015

Transferible	Si

Atributo	Colaborador
Nombre	Carlos Caballero
Salario	76.000€
Puesto de trabajo	Ojeador de la cantera

Atributo	Colaborador
Nombre	Luis González
Salario	Unknown
Puesto de trabajo	Encargado de finanzas