Nombre y Apellidos:

Instrucciones. La tarea constará de diferentes puntos que crearan un modelo de clases y un programa que desarrollaremos en la unidad siguiente.

La tarea se entregará en el aula virtual en un ZIP **tareaUnidad6NombreApellidos.Zip** conteniendo los ejercicios contestados. Crear un proyecto en eclipse **tareaTema3NombreApellidos**. Cada ejercicio en Java se guardará en una clase diferente como **Ejercicio1NombreApellidos.java** si es el Ejercicio 1, etc.

Resultados de aprendizaje

RAE 4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

Criterios de evaluación:

f) Utilizar mecanismos para controlar la visibilidad de las clases y de sus miembros. Indicadores 35, 42,43

g) Definir y utilizar clases heredadas. Indicadores 36, 42, 43

h) Crear y utilizar métodos estáticos. Indicadores 37, 42, 43

i) Definir y utilizar interfaces. Indicadores 38, 42, 43

Detalles de la tarea de esta unidad.

Enunciado.

En esta unidad has visto las características fundamentales de la programación funcional en java 8

Para poder realizar la tarea de esta unidad vas a crear un proyecto y una clase por ejercicio.

También tendrás que realizar lecturas por teclado y visualizaciones por pantalla.

Crea un proyecto Java de nombre Tarea**6NombreApellidos**. Y para **cada ejercicio una clase EjercicioX.java**

Nombre y Apellidos:

Ejercicio 1. Expresiones lambda simples. (2 puntos)

Crea una clase Ejercicio1.java y añadidle **una función main**. Definir un interfaz funcional propio **MiInterfazFuncional<Double> interno con un método abstracto** prueba Lambda: `public Double pruebaLambda(Double valor)` que se definirá de esta manera. Realizar entonces los siguientes pasos:

Indicadores 35, 36, 37, 38, 42, 43

1. Pedir un número decimal por pantalla.
2. Escribid una expresión lambda que calcule el inverso o reciproco de un número + 1. Probadla con el interfaz funcional. Escribid el resultado por pantalla **(0,5 puntos)**
3. Escribid una expresión lambda que calcule el cubo de un número dividido por (numero +4) . Probadla con el interfaz funcional. Escribid el resultado por pantalla **(0,7 puntos)**
4. Escribid una expresión lambda que calcule el cubo de un número dividido por (numero +4) . Probadla con el interfaz funcional. Escribid el resultado por pantalla
   (0,8 puntos)

   El resultado de la ejecución será:

```
Escriba un número decimal
6,7
Resultado de reciproco + 1 es: 1.1492537313432836
Resultado cubo de 6.7 dividido por 6.7 más 428.108691588785053
Resultado cuadrado de 6.7 dividido por el aleatorio 0.17113397534682417 es:
45.380895511459904
```

Ejercicio 2. Expresiones lambda de bloque. (2 puntos)

Crea una clase **Ejercicio2.java** y añadidle una función main. Definir un interfaz funcional propio **MiInterfazFuncionalBloque para enteros** interno con un método abstracto prueba lambda: `pruebaLambdaBloque` que se definirá de esta manera. Realizar entonces los siguientes pasos:

Nombre y Apellidos:

Indicadores 35, 36, 37, 38, 42, 43

1. **Pedir un número entero** por pantalla **(0,5 puntos)**
2. **Escribid una expresión lambda** de bloque con un bucle while que escriba la lista de los n primeros números pares, calcule la suma de los n primeros números pares y la muestre por pantalla. **(0,7 puntos)**
3. **Escribid una expresión lambda que escriba los números primos entre 1 y n que acaben en 3**. Usad un bucle for. **(0,8 puntos)**

El resultado de la ejecución será:

```
Escriba un número entero
50
La serie de los 50 primeros numero pares es: 0, 2, 4, 6, 8, 10, 12, 14, 16,
18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54,
56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92,
94, 96, 98,
 Y la suma total los 50 primeros numeros pares es: 2500
La serie de de los 50 primeros numeros primos que acaban en 3 es: 3, 23, 43,
83, 103, 163, 223, 263, 283, 383, 443, 463, 503, 523, 563, 643, 683, 743,
823, 863, 883, 983, 1063, 1103, 1123, 1163, 1223, 1283, 1303, 1423, 1483,
1523, 1543, 1583, 1663, 1723, 1783, 1823, 2003, 2063, 2083, 2143, 2203, 2243,
2383, 2423, 2503, 2543, 2663,
 Y la suma total de los 50 primeros numeros primos es 57747
```

Nombre y Apellidos:

Ejercicio 3. Interface Supplier y Predicate. (2 puntos)

Crea una clase **Ejercicio3.java** y añadidle una función main. En esta función vamos a definir varios interfaces predefinidos

Indicadores 35, 36, 37, 38, 42, 43

1. Escribid un Supplier **aleatorioCincuenta** de tipo predefinido Integer que nos proporcione un número aleatorio entre 1 y 50.
2. Escribid un Predicate **divisiblePorCinco** de tipo predefinido Integer que nos diga si el número es divisible por 5. Aplicarlo a la salida del Supplier. **(0,5 puntos)**
3. Escribid un Predicate MenorQue30 de tipo predefinido Integer que nos diga si el número es mayor que 5 y menor que 30. Aplicarlo a la salida del Supplier **AleatorioCincuenta**. **(0,7 puntos)**
4. Crear un nuevo Predicate CombinaDos que combine los dos anteriores con un and. Aplicarlo a la salida del Supplier **AleatorioCincuenta**. **(0,8 puntos)**

El resultado de la ejecución será:

```
Numero supplier aleatorioCincuenta 20 es divisible por 5

Numero supplier aleatorioCincuenta 20 es mayor que 5 y menor que 30

Numero supplier aleatorioCincuenta 20 es mayor que 5 y menor que 30 y
divisible por 5


Numero supplier aleatorioCincuenta 31 no es divisible por 5

Numero supplier aleatorioCincuenta 31 no es mayor que 5 y menor que 30
```

Nombre y Apellidos:

```
Numero supplier aleatorioCincuenta 31 no es mayor que 5 y menor que 30 o
divisible por 5
```

Ejercicio 4. Interface Supplier y Predicate. (2 puntos)

Indicadores 35, 36, 37, 38, 42, 43

Crea una clase **Ejercicio4.java** y añadidle una función main. En esta función vamos a definir varios interfaces predefinidos.

1. Escribid un **Supplier numeroPorPantalla** de tipo <Integer> que nos devuelva un numero leído por pantalla con una expresión lambda de bloque. **(1 punto)**
2. Escribid un **Predicate esPrimo** que nos diga si el numero es primo con una expresión lambda de bloque que tenga un bucle for. Aplicarlo al resultado de del supplier numeroPorPantalla al predicate. **(1 punto)**

El resultado de la ejecución sería:

```
Escriba un número entero por pantalla
24
El numero generado por el supplierPantalla 24 no es primo

Escriba un número entero por pantalla
23
El numero generado por el supplierPantalla 23 es primo
```

Ejercicio 5. Interface Supplier, Function, Consumer. (2 puntos)

Indicadores 35, 36, 37, 38, 42, 43

Nombre y Apellidos:

Crea una clase **Ejercicio5.java** y añadidle una función main. En esta función vamos a definir varios interfaces predefinidos.

1. Usamos el interface Supplier **numeroPorPantalla** del ejercicio anterior y lo transformamos para que recoja un decimal. Llamadolo **numeroPorPantalla (0,4 puntos)**
2. Creamos un interface Function funcionTrigo para decimales para calcular la función $(seno(x)^2 + cos(x)^3)/5$. Usad la librería Math para realizar para las potencias,los senos y los cosenos. **(0,6 puntos)**
3. Creamos un interface Function cifrasNumero para decimales que nos devuelva un entero con el número de cifras enteras tiene un número con la instrucción do While.
   (0,6 puntos)

4. Crear un Consumer escribePantalla que escriba por pantalla los resultados. Recibirá un <String> como entrada. **(0,4 puntos)**

El resultado de la ejecución sería:

```
Escriba un número decimal por pantalla
2456
El resultado de aplicar la función trigonométrica al numero 2456.0 es
0.17180821904080595
El numero total de cifras del numero 2456.0 es 4
```