Programación 2 - 2023

Práctico 1 - Genéricos y Excepciones

Ejercicio 1

Crear una clase Empleado con atributo nombre, apellido, legajo, aniosTrabajados
Sobrescribir el método toString() para ver una representación entendible de un objeto de la clase
Crear en la clase de arranque un Set de empleados.
Crear 5 objetos empleado, donde 2 empleados tienen los mismos valores de atributos, agregarlos al set
Iterar sobre el set y mostrar los objetos empleados
Crear una herencia de la clase Empleado (EmpleadoSet) y modificarla para que cuando se agreguen EmpleadoSet a un set, los repetidos sean ignorados. Investigar como se hace.
Crear 5 objetos empleadoSet, donde 2 empleados tienen los mismos valores de atributos, agregarlos al set
Iterar sobre el set y mostrar los objetos empleados. Verificar que el duplicado no existe.

Ejercicio 2

Crear una clase Empleado con atributo nombre, apellido, legajo, aniosTrabajados
Crear una clase genérica ListaGenerica que permita mantener una lista de objetos que implementan la interfaz comparable.

1. Definir un método que permita agregar un elemento a la lista.
2. Definir un método que devuelva la lista de objetos.
3. Definir un método que devuelva la cantidad de elementos almacenados en la lista.
4. Definir un método que permita agregar un elemento al principio de la lista.
5. Definir un método que permita ordenar la lista.
6. Definir un método que permita desordenar (barajar) la lista.
7. Definir un método que permita almacenar un elemento en una posición particular de la lista, los parámetros que recibe son el objeto a almacenar y la posición en la lista.
8. Definir un método que devuelva el objeto en una posición específica de la lista
9. Definir un método que devuelva el objeto en la primer posición de la lista
10. Definir un método que devuelva el objeto en la ultima posición posición de la lista
11. Definir un método que remueva un objeto en una posición particular de la lista.

Ejercicio 3

Mejorar al ejercicio anterior agregando las siguientes excepciones:

- Una excepción para la función 7 si el índice es más grande que la cantidad de elementos de la lista.
- Una excepción para la función 8 si el índice es mayor a la cantidad de elementos en la lista.
- Una excepción para la función 9 y 10 si la lista de elementos está vacía.

- Una excepción para la función 11 si el elemento en la lista no existe.

El código debe incluir una parte donde se generan a propósito cada una de las excepciones definidas y el tratamiento del mismo.

Ejercicio 4

Crear una interfaz llamada FiguraGeometrica con los métodos calcularPerimetro y calcularSuperfice.
Crear 3 clases que implementen la interfaz FiguraGeometrica. Pueden ser círculo, triángulo, rectángulo, cuadrado, pentágono.
Crear una lista para almacenar figuras geométricas y agregar 5 elementos.
Recorrer la lista mostrando información de cada uno de los objetos (recomendable que las clases prescriban el método toString) y mostrar el perímetro y la superficie de cada uno.

Ejercicio 5

Modificar el ejercicio 4 y agregar 2 parámetros más, uno para la ubicación en el eje X y otro para el eje Y.
Agregar excepciones para que:

- Al crear el objeto los valores de los parámetros no pueden ser negativos. Los parámetros a contemplar deben ser los que definen a la figura geométrica. También debe verificarse al querer cambiar los parámetros con los setter.
- Al crear el objeto los valores de los parámetros de la ubicación en eje X y eje Y no pueden ser negativos. También debe verificarse al querer cambiar los parámetros con los setter.

El código debe incluir una parte donde se generan a propósito cada una de las excepciones definidas y el tratamiento del mismo.

Ejercicio 6

Crear una clase Empleado con atributo nombre, apellido, legajo, aniosTrabajados
Sobrescribir el método toString() para ver una representación entendible de un objeto de la clase.

Crear una lista con 5 empleados.
Crear un mapa <String, Empleado> y copiar los 5 empleados, la clave del mapa debe ser un string con la combinación de apellido+",”+nombre.

Iterar sobre el mapa y mostrar la clave y el objeto Empleado.

Ejercicio 7

Crear una clase Empleado con atributo nombre, apellido, legajo, aniosTrabajados

Programación 2 - 2023

Sobrescribir el método toString() para ver una representación entendible de un objeto de la clase.

Agregar un método estático que reciba un string con formato “nombre=nombre, apellido=apellido, legajo=legajo,aniosTrabajados=aniosTrabajados” y que devuelva un nuevo objeto del tipo Empleado.
Ejemplo:
“nombre=Pablo, apellido=Marquez, legajo=E001, aniosTrabajados=10”

El string puede estar separado por coma, con o sin espacio y se debe contemplar en caso de tener espacio que no quede reflejado en el valor de los atributos, los nombres de los atributos pueden estar desordenados y estar en mayúsculas, minúsculas o mezclado.

Todos estos ejemplos deberían generar el mismo objeto:
“nombre=Pablo, apellido=Marquez, legajo=E001, aniosTrabajados=10”
“nombre=Pablo,apellido=Marquez,legajo=E001,aniosTrabajados=10”
“nombre=Pablo ,apellido=Marquez ,legajo=E001, aniosTrabajados=10”
“Nombre=Pablo,apellidO=Marquez ,LEGAJO=E001, ANIOStrabajados=10”
“nombre=Pablo,legajo=E001,aniosTrabajados=10,apellido=Marquez”

Crear una excepción al crear el objeto que evalúe las siguientes condiciones:

- todos los campos tienen que estar presentes
- todos los campos tienen que tener datos
- el campo aniosTrabajados tiene que ser un valor numérico entero, mayor a cero

Cuando nos referimos a que todos los campos tienen que tener valor se refiere a que no puede haber una definición de este tipo:
“nombre=Pablo, apellido=, legajo=E001, aniosTrabajados=10”

El código debe incluir una parte donde se generan a propósito cada una de las excepciones definidas y el tratamiento del mismo.