Ejercicio Código 4

Autor: José Guillermo Sandoval Huerta

Fecha: 03 noviembre 2023

Uso Predicate y Lambdas

La interfaz Predicate se emplea para representar un predicado, es decir, una función que toma un argumento y devuelve un valor booleano (true o false). Esta interfaz tiene un único método abstracto llamado test, que toma un argumento y devuelve un valor booleano. La firma del método test es la siguiente:

- **boolean test(T t);**

Mientras que una lambda es una expresión anónima que permite definir y crear instancias de interfaces funcionales de manera concisa, se utilizan para simplificar la implementación de interfaces funcionales.

Su sintaxis es la siguiente:

- **(parámetros) -> expresión o bloque de código;**

Mencionado lo anterior se implementó un programa con temática de Piratas, donde cada objeto instanciado tiene ciertos atributos para que por medio de métodos implementados logremos saber su habilidad del uso de armas y su rango. A continuación, se explica las clases creadas.

Código 1. Pojo Pirata.

```
…

public class Pirata {
   private String nombre;
   private String apodo;
   private String arma;
   private int tesoros;

   public Pirata(String nombre, String apodo, String arma, int tesoros) {
                this.nombre = nombre;
                this.apodo = apodo;
                this.arma = arma;
                this.tesoros = tesoros;
        }
   …
}
```

Clase con 4 atributos (nombre, apodo, arma, tesoros).

Código 2. Clase Principal.

```
…

public class Principal{
	public static void main(String[] args) {

		List<Pirata> listaPiratas = new ArrayList<>();
		listaPiratas.add(new Pirata("Edw ard Teach", "Barbanegra", "Espada recta",12));
		listaPiratas.add(new Pirata("Bartholomew Roberts", "El Diablo Negro", "Pistolas de chispa",9));
		listaPiratas.add(new Pirata("Henry Morgan", "El Pirata Morgan", "Sable",17));

		// Tipo Arma
		poderArma(listaPiratas, pir -> pir.getArma().length() > 5);	// Uso lambda
		System.out.println();

		// Rango Pirata
		rango(listaPiratas, pir -> pir.getTesoros()>10);	// Uso lambda
	}

	private static void poderArma(List<Pirata> listaPiratas, Predicate<Pirata> pre) {
		for(Pirata pirata:listaPiratas) {
			if(pre.test(pirata))	// Uso de Predicate
				System.out.println("Poder: " + pirata.getArma() + " - Efectiva");
			else
				System.out.println("Poder: " + pirata.getArma() + " - Debil");
		}
	}

	private static void rango(List<Pirata> listaPiratas, Predicate<Pirata> pre) {
		for(Pirata pirata:listaPiratas) {
			if(pre.test(pirata))	// Uso de Predicate
				System.out.println("Gran Pirata: " + pirata.getApodo());
			else
				System.out.println("    Pirata: " + pirata.getApodo());
		}
	}
}
```

Donde en el método main, se instancia una lista de objetos de tipo Pirata y se implementan dos métodos para conocer la habilidad del uso de su arma y el rango pirata. Cada uno manda a llamar al método test de la interfaz Predicate que recibe una lambda, la cual define el resultado de su comportamiento.

- **Lambda 1**: pir -> pir.getArma().length() > 5;

  De acuerdo al nombre del arma que es un string, si este no tiene un tamaño mayor a 5, su es débil, caso contrario su uso es efectivo.

- **Lambda 2**: pir -> pir.getTesoros()>10;

  De acuerdo al número de tesoros encontrados, si consiguieron un número mayor a 10, se les otorga el rango de Gran Pirata, en caso contrario solamente se quedan como Piratas.