LAB – 8 – HOMEWORK

Java Functional Interface API link:

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/function/package-summary.html

Problem 1: Identify the suitable functional Interface from the Java API and complete code for the given tasks. Refer to the Problem1.java file for the startup code.

Problem 2: Get practice on how sorting performed using Inner classes and how can easily do it with Lambdas by doing solving this problem.

```
class Product {
	final String title;
	final double price;
	final int model;

	public String getTitle() {
		return title;
	}

	public double getPrice() {
		return price;
	}

	public int getModel() {
		return model;
	}

	public Product(String title, Double price, int model) {
		this.title = title;
		this.price = price;
		this.model = model;
	}

	@Override
	public String toString() {
		return String.format("\n %s : %s : %s", title, price, model);
	}
}
```

a. Sort by implementing a comparator for price attributes and printing the product list using separate comparators outside the class.
b. Sort by implementing a comparator for title attribute and print product list using separate comparators outside the class.
c. Implement the sort method so that only one type of Comparator is used for tasks a & b by referring comparator3 package.
d. If the title is the same, use the model as another attribute to sort it. Do this by using lambdas. (Java Functional way of using Lambda by referring closures.java8 package)

Get practice to use method references

3. In the lecture, one of the examples of a method reference of type *object::instanceMethod* was this::equals. Since every lambda expression must be converted to a functional interface, find a functional interface in the java.util.function package that would be used for this lambda expression. The implicit reference `this' refers to the active object.

In which you have to reference this::equals with an appropriate type(Suitable functional Interface), implement a method myMethod(MyClass cl) [testing method to check the equality] which uses this method expression to return true if cl is equal to 'this'.

Startup Code

```
public class MyClass {

        int x ;
        String y;
        public MyClass(int x, String y) {
                this.x = x;
                this.y = y;
        }
// testing method to check the equality
     public void myMethod(MyClass cl) {
        //Implement
                System.out.println(this.equals(c1));
        }

        @Override
        public boolean equals(Object ob) {
        if(ob == null) return false;
        if(ob.getClass() != getClass()) return false;
```

```
        MyClass mc = (MyClass)ob;
        return mc.x == x && mc.y.equals(y);
        }

        public static void main(String[] args) {
                MyClass myclass = new MyClass(1, "A");
                MyClass myclass1 = new MyClass(1,"B");
                myclass.myMethod(myclass);  //
                myclass.myMethod(myclass1);

        }

}
```

Problem 4:

```
String[] names = {"Alexis", "Tim", "Kyleen", "Kristy"};
```

a. Use Arrays.sort() to sort the names by ignore case using Method reference.
b. Convert the sorted names array into List.
c. Print the sorted list using method reference.