sOFTWARE 2 PRACTICAL

## Comparable & Comparator

Week 08 – Practical 8

For this practical, you should create a Java project. Within the project, create a package shopping, and within this package two classes, Laptop and Inventory. The aim of the exercises is to be able to sort a list of laptops by different field such as price, RAM or brand.

### Exercise 1: *The* Laptop *class*

### Implement the class Laptop representing a model of a laptop. An instance of the class Laptop has:

### the following attributes with package access modifier.

* a String barcode,
* a String brand,
* a String model containing for example “surface book 2”,
* an int ram containing the ram in Gb,
* an int ssd containing the SSD size in Gb,
* an int price containing the price of the laptop in pence.

1. You should provide adequate constructor, setter (if any), getters (if any) and toString() method.
2. The class has 5 String constants BRAND, MODEL, RAM, SSD, and PRICE. Their respective values do not matter as we should use their names instead. These constant will be used later to create comparator based on the field they represent.
3. the following public methods

* public static Comparator<Laptop> getComparator(String field) which returns a comparator object sorting laptop by the field represented by one of the constant define in (3).   
  The method throws an IllegalArgumentException if the parameter field is not one of the constants defined earlier.

### Exercise 2: *The* Inventory *class*

Implement the class Inventory representing the list of laptops currently listed on our shopping app. The class has at least:

1. the following attribute with package access modifier,

* a Map<String, Laptop> stockList representing list of laptops currently listed on our shopping app indexed by their barcode (the keys of the map).

1. public constructor Inventory() that initialise the inventory to an empty map.,
2. and public methods:

* public void addLaptop(Laptop laptop) which adds a laptop to the stockList,
* public List<Laptop> sortInventory(String field) which returns the list of laptops listed in the stockList sorted in ascending order using the field passed in parameter.   
  The field value should be one of the constants defined in the class Laptop, if that is not the case, the method throws an IllegalArgumentException.

Of course you can add more methods if you wish to do so.

### Exercise 3:

Putting it all together,

public static void main(String[] args) throws Exception {

  Inventory inventory = new Inventory();

  inventory.addLaptop(new Laptop("001", "Apple", "mac", 8, 128, 60000));

  inventory.addLaptop(new Laptop("002", "Microsoft", "SurfaceBook2",

                                16, 512, 260000));

  inventory.addLaptop(new Laptop("003", "HP", "Sensa", 8, 256, 80000));

  inventory.addLaptop(new Laptop("004", "Asus", "zen13", 16, 256, 100000));

  inventory.addLaptop(new Laptop("005", "Asus", "minibook", 2, 64, 40000));

  System.out.println(inventory.sortInventory(Laptop.BRAND));

  System.out.println(inventory.sortInventory(Laptop.PRICE));

}

the program should have an output like:

[Laptop<Apple:mac(RAM=8, SSD=128, Price=£600.0)>,   
Laptop<Asus:zen13(RAM=16, SSD=256, Price=£1000.0)>, Laptop<Asus:minibook(RAM=2, SSD=64, Price=£400.0)>,   
Laptop<HP:Sensa(RAM=8, SSD=256, Price=£800.0)>, Laptop<Microsoft:SurfaceBook2(RAM=16, SSD=512, Price=£2600.0)>]

[Laptop<Asus:minibook(RAM=2, SSD=64, Price=£400.0)>,   
Laptop<Apple:mac(RAM=8, SSD=128, Price=£600.0)>,   
Laptop<HP:Sensa(RAM=8, SSD=256, Price=£800.0)>,   
Laptop<Asus:zen13(RAM=16, SSD=256, Price=£1000.0)>, Laptop<Microsoft:SurfaceBook2(RAM=16, SSD=512, Price=£2600.0)>]

### Exercise 4: *Reflection*

How would you refactor your code so a user can choose to have the laptops sorted in increasing or decreasing order?