Zu 36.4 – Weitere generische Typen

Unterschied Comparable<T> und Comparator<T>

Erklärvideo (nicht meins) mit diesen Beispielen:
https://www.youtube.com/watch?v=oAp4GYprVHM

Anbei die beiden Klassen aus dem Video. Main() ist in der Klasse Runner.

printByRam(laps) nutzt Comparable
printByPrice(laps) nutzt Comparator

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
//_________________ComparATOR_____________________
import java.util.List;


public class Runner {



	public static void printByRam(List<Laptop> list){
	//_________________ComparABLE_____________________
	//nutzt	public class Laptop implements Comparable<Laptop>{
	//und		public int compareTo(Laptop lap2) {
	//in Klasse Laptop

		Collections.sort(list);

		for (Laptop l : list) {
			System.out.println("Nach Ram sortiert: " + l);
		}
	}




	//_________________ComparATOR_____________________
	public static void printByPrice(List<Laptop> list){
		//
		//
		//_________________ComparATOR_____________________
		//kann in dieser Klasse oder ohne Methode im main() implementiert werden
		Comparator<Laptop> myComp = new Comparator<Laptop>() {

			//compare(Xy x, Xy y) gehört zu ComparATOR und muss implementiert
			werden, Name der Methode kann beliebig Gewählt werden
			public int compare(Laptop lap1, Laptop lap2) {
				if(lap1.getPrice() > lap2.getPrice()) {
					return 1;
				}else if(lap1.getPrice() < lap2.getPrice()) {
					return -1;
				}
				return 0;
```

```
                }
        };//____<---__ ; ______!!!!!
        //!!!!!!!!!!!!!!!!!!!!!!!!!!


        Collections.sort(list, myComp);
        // 2 Parameter !!!!!!!!! nutzt myComp

        for (Laptop l : list) {
                System.out.println("Nach Preis sortiert: " + l);
        }
    }


    public static void main(String[] args) {

        List<Laptop> laps = new ArrayList<>();
        // zweites <> übernimmt <Laptop> aus vorderem Teil
        // --> List<Laptop> laps = new ArrayList<Laptop>();

        laps.add(new Laptop("Bell", 16, 800));
        laps.add(new Laptop("Abble", 8, 1200));
        laps.add(new Laptop("Pacer", 12, 700));


        printByRam(laps);
        printByPrice(laps);

    }

}



public class Laptop implements Comparable<Laptop>{
                                //_______________ComparABLE_____________________

    private String brand;
    private int ram;
    private int price;



    //
    //in Eclipse rechte Taste / Source / Generate Constructor using fields
    //
    public Laptop(String brand, int ram, int price) {
        super();
        this.brand = brand;
        this.ram = ram;
        this.price = price;
    }


    //
    //in Eclipse rechte Taste / Source / Generate Getter & Setter
    //
```

```
	public String getBrand() {
		return brand;
	}
	public void setBrand(String brand) {
		this.brand = brand;
	}
	public int getRam() {
		return ram;
	}
	public void setRam(int ram) {
		this.ram = ram;
	}
	public int getPrice() {
		return price;
	}
	public void setPrice(int price) {
		this.price = price;
	}

	//
	//in Eclipse rechte Taste / Source / Generate toString()
	//
	@Override
	public String toString() {
		return "Laptop [brand=" + brand + ", ram=" + ram + ", price=" + price +
"]";
	}



	@Override
	public int compareTo(Laptop lap2) {
	//_________________ComparABLE_____________________
	//gehört zu Comparable und muss implementiert werden

		// this > lap2 = +;
		// this < lap2 = -;
		// this == lap2 = 0;

		if(this.getRam() > lap2.getRam()) {
			return 1;
		}else if (this.getRam() < lap2.getRam()) {
			return -1;
		}
		return 0;
	}




}
```

Die Beispiele zu HashSet und HashMap aus dem Übungsheft

```
import java.util.HashSet;
import java.util.Set;


public class Hash_Set {
```

```
	//Fehleralarm: wenn man die Klasse HashSet nennt überschreibt man die java.util-
	Klasse und Zeile 12 produziert einen Fehler

	public static void main(String[] args) {

		Set<String> menge = new HashSet<String>();
		String a = "Hallo Welt!";
		String b = "Hallo";
		String c = "Welt!";
		String d = b + " " + c;
		System.out.println("menge.add(a): " + menge.add(a)); //liefert true
		System.out.println("menge.add(b): " + menge.add(b)); //liefert true
		System.out.println("menge.add(c): " + menge.add(c)); //liefert true
		System.out.println("menge.add(d): " + menge.add(d)); //liefert false, da
                                    gleiche Zeichenkette schon enthalten ist

		System.out.println("menge.size(): " + menge.size()); //liefert 3
		System.out.println("menge.contains(b): " + menge.contains(b)); //liefert
                                                                    true
		System.out.println("menge.contains(\"Hallo\"): " +
	menge.contains("Hallo")); //liefert true
		System.out.println("menge.contains(\"Hello\"): " +
	menge.contains("Hello")); //liefert false

		System.out.println("menge.remove(d): " + menge.remove(d)); //zu löschender
                                                Wert enthalten, liefert true
		System.out.println("menge.size(): " + menge.size());//liefert 2 zurück

	}

}


import java.util.Map;

public class Hash_Map {

	public static void main(String[] args) {


		Map<Integer, String> myMap = new HashMap<Integer, String>();
		String a = "Hallo";
		String b = "Welt!";

		//.put
		System.out.println("myMap.put(10 , a): " + myMap.put(10, a)); //liefert
		null, da vorher kein Wert zu 10 gespeichert war
		System.out.println("myMap.put(12 , b): " + myMap.put(12, b)); //liefert
		null, da vorher kein Wert zu 12 gespeichert war

		//.get
		System.out.println("myMap.get(10): " + myMap.get(10));// liefert Hallo
		System.out.println("myMap.get(4): " + myMap.get(4));// liefert null

		System.out.println("myMap.size(): " + myMap.size());// liefert 2
		System.out.println("myMap.remove(12): " + myMap.remove(12));// liefert
                                                                    "Welt"
	}
}
```