09Uebung
1 5/5

```
package uebung091;

public class Auto {

		private String name;
		private Reifen[] reifen;
		private Motor motor;
		private Kofferraum kofferraum;
		private int sitzanzahl;
		private double maxkmh;
		private int tueranzahl;

		 public Auto(int anzahl) {
			if( anzahl >= 3 && anzahl <= 5) {
			reifen = new Reifen[anzahl];
			for(int i = 0; i < anzahl; i++) {
				reifen[i] = new Reifen();
			}
			}

		 }



		public Reifen[] getReifen() {
			return reifen;
		}

		public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSitzanzahl() {
        return sitzanzahl;
    }

    public void setSitzanzahl(int sitzanzahl) {
        this.sitzanzahl = sitzanzahl;
    }

    public double getMaxkmh() {
        return maxkmh;
    }

    public void setMaxkmh(double maxkmh) {
        this.maxkmh = maxkmh;
    }

    public int getTueranzahl() {
        return tueranzahl;
    }

    public void setTueranzahl(int tueranzahl) {
        this.tueranzahl = tueranzahl;
    }
```

```
        // Methode ersetzen
        public boolean isfaster(Auto other) {
            if(this.getMaxkmh() > other.getMaxkmh()) {
                return true;
            }

            else {
                return false;
            }
        }

        public void replacereifen(Reifen other, int index) {
            if(this.getReifen() != null) {
            for(int i = 0; i < reifen.length; i++ ) {
                if(i == index) {
                    reifen[i] = other;
                }
            }
            }
        }

}

package uebung091;

public class Kofferraum {

    private double kofferaumqcm;
```

```
	public double getKofferaumqcm() {
		return kofferaumqcm;
	}

	public void setKofferaumqcm(double kofferaumqcm) {
		this.kofferaumqcm = kofferaumqcm;
	}


}
package uebung091;

public class Motor {
	private double hubraumqcm;

	public double getHubraumqcm() {
		return hubraumqcm;
	}

	public void setHubraumqcm(double hubraumqcm) {
		this.hubraumqcm = hubraumqcm;
	}


}


package uebung091;
```

```java
public class Reifen {

	private double profilgütemm;
	private boolean platt = false;



	public double getProfilgütemm() {
		return profilgütemm;
	}
	public void setProfilgütemm(double profilgütemm) {
		this.profilgütemm = profilgütemm;
	}
	public boolean isPlatt() {
		return platt;
	}
	public void setPlatt(boolean platt) {
		this.platt = platt;
	}



}
```

2 6,5/7

Ausgabe nicht ganz korrekt

```
package uebung092;

public class LinkedIntListElement {

	private LinkedIntListElement next;
	private Integer value;


	//GETTER SETTTER
	public LinkedIntListElement getNext() {
		return next;
	}

	public void setNext(LinkedIntListElement next) {
		this.next = next;
	}

	public Integer getValue() {
		return value;
	}

	public void setValue(Integer value) {
		this.value = value;
	}
}
```

```
package uebung092;
```

```
public class SortedLinkedIntList {

	private LinkedIntListElement start;
	private int size = 0;
	private LinkedIntListElement iterCurrent;


	public int size() {
		return size;
	}


	public LinkedIntListElement getStart() {
		return start;
	}


	public void setStart(LinkedIntListElement start) {
		this.start = start;
	}


	//GET
	public int get(int index) {
		if (start == null) { // list is empty
			return -1;
		  }
```

```
        LinkedIntListElement current = start;
        int pos = 0; // counter for finding the right position
            while (pos < index) {
            if (current.getNext() == null) {
            // list does not have enough elements
                    return -2;
          }
          current = current.getNext();
          pos++;
      }
      return current.getValue();
      }

  //Methoden

  public void add(int value) {
        LinkedIntListElement elem = new LinkedIntListElement();
        elem.setValue(value);
        if (start == null) { // list is empty
            start = elem;

            size++;
        }
        else {
        LinkedIntListElement current = start; //WIRD HIER NEUES ELEMENT
"CURRENT" ERZEUGT????
        while (current.getNext() != null && current.getNext().getValue() < value)
{
            // find last element & test if next element < than current
            current = current.getNext();
```

nein - es wird eine Variable angelegt, die ein (bereits existierendes) Element (start) speichert. Dieses LinkedIntListElement-Objekt wird jetzt also 2x gespeichert, es bleibt aber dasselbe Objekt. Kurz: current.equals(start) würde true zurückgeben.

Ein neues Element würde man über new ...() erzeugen. In dem Fall würde equals false zurückliefern (auch wenn z.B. beide den gleichen Wert speichern), da es sich um zwei verschiedene Objekte handelt.

```
        }
    if(current.getNext() == null) {
    current.setNext(elem);

    size++;
    }

    //Wenn nächster Wert größer ist als value wird liste erweitert
else if(current.getNext().getValue() > value) {
ansonsten gibt es hier Probleme, wenn getNext null ist
    // größeres Element wird tmp gespeichert
        LinkedIntListElement tmp = current.getNext();
        // value wird beim nächsten gestezt
        current.setNext(elem);
        // ein schritt weiter
        current = current.getNext();
        //tmp größter value wird hinten drangehangen + liste um 1 erweitert
        current.setNext(tmp);
        size++;
    }

    /*//Wenn Startelement größer ist als Value
else if(current.getValue() > value) {
- es gibt (wenn die Liste nicht leer ist) insg. 3 verschiedene Fälle, von denen immer nur einer eintreten kann (am Ende einfügen, zwischen 2 Elementen einfügen, am Anfang einfügen)
        LinkedIntListElement tmp = current;
        current = elem;
        current.setNext(tmp);
        size++;
    }*/
```

```
        }
}

public void reset() {
        iterCurrent = start;
}

public Integer getNext() {
        Integer result = iterCurrent.getValue();
        iterCurrent = iterCurrent.getNext();
        return result;
}

public boolean hasNext() {
        return iterCurrent != null;
}

public int[] toArray() {
        this.reset();
        int[] array1 = new int[this.size()];
        if(this.hasNext()) {  das ist nicht nötig, wird ja quasi automatisch in der for Schleife geprüft
                for(int i = 0; i < this.size(); i++) {
                        array1[i] = this.getNext();
                }
        return array1;
        }

        else {
                return null;
        }
```

```
        }

}
```

Unit-Test fehlt. Bitte allen Code in die pdf einfügen, ich schau mir nicht immer die Codedateien an.

5/8

3

Clear nicht gemacht??

```
package uebung093;



public class Container {

	private ContainerElement start;
	private int maxWeight = 0;
	private int size = 0;
	private int weight = 0;

	//Konstruktor
	public Container(int maxWeigth) {
		int mw = maxWeigth;
		setMaxWeight(mw);
		}

	//GETTER SETTER
	public int getMaxWeight() {
		return maxWeight;
	}

	private void setMaxWeight(int value) {
		maxWeight = value;
	}
```

```
//METHODEN
//Warum nicht über get??
public int size() {
        return size;
}

public int getWeight() {
        ContainerElement current = start;
        if(current == null) {
                return 0;
        }
        int tmpweight = 0;
        for(int i = 0; i < this.size(); i++) {
                tmpweight += current.getValue().getWeight();
                current = current.getNext();
        }
        return tmpweight;



}

public void clear() {
        //setSize() = 0;
          size = 0;
          weight = 0;
          start = null;
}

public boolean add(Package value) {
        int freespace;
        freespace = (this.getMaxWeight() - this.getWeight());
        if(freespace >= value.getWeight()) {
```

Weil size in der Lösung nicht als Attribut vorgesehen ist.

Außerdem werdet ihr später keine Listen mehr selber implementieren müssen, sondern Implementierungen von List (https://docs.oracle.com/javase/8/docs/api/java/util/List.html) nutzen. Und dort heißt die entsprechende Methode auch size(), sodass ihr euch schonmal die "richtigen Methodenaufrufe" angewöhnen könnt. ;)

weight wird doch als Attribut gespeichert. Das könnte man bei jedem Einfügen aktualisieren und müsste dann hier nicht über die Liste iterieren.

s.o., weight wird nicht aktualisiert

```
                //add package
            ContainerElement elem = new ContainerElement();
            elem.setValue(value);
            if (start == null) { // list is empty
                start = elem;
              }
            else {
            ContainerElement current = start;
            while (current.getNext() != null) { // find last element
                    current = current.getNext();
                }
            current.setNext(elem);
            }
            size++;
            return true;
            }
            else {
                    return false;
            }
        }
          addOpt fehlt
}


package uebung093;

public class ContainerElement {

        private ContainerElement next;
        private Package value;
```

```
    public ContainerElement() { }

    public ContainerElement(ContainerElement next, Package value) {
        this.next = next;
        this.value = value;
    }

    public ContainerElement getNext() {
        return next;
    }

    public void setNext(ContainerElement next) {
        this.next = next;
    }

    public Package getValue() {
        return value;
    }

    public void setValue(Package value) {
        this.value = value;
    }

}
```

```
package uebung093;
public class Package {
```

```
    private int weight;

    public Package(int weight) {
        this.weight = weight;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

}
```

16,5/20