12 – Verständnisfragen zur selbstständigen Beantwortung

(1) Erläutern Sie die Konzepte boxing und unboxing in Java anhand eines Beispiels.

(2) Seit Java 9 gibt es den automatischen Caching-Mechanismus für Wrapperklassen. Recherchieren Sie, was es damit auf sich hat und wie die folgende Codezeile jetzt erzeugt werden sollte:

```
Boolean   b  = new Boolean(true);
```

(3) Angenommen, wir wollen verschiedene Sortieralgorithmen zum Sortieren von Zahlen in einem Array implementieren und die Zeit vergleichen:

```
int[] liste = {1,5,3,8,9,6,7,4,2};
int zeitA = sortierMethodeA(liste);
int zeitB = sortierMethodeB(liste);
int zeitC = sortierMethodeC(liste);
int zeitD = sortierMethodeD(liste);
```

Worauf müssen wir bei der Implementierung der Methoden hier besonders achten?

(4) Erläutern Sie den Unterschied zwischen einer Objektreferenz und einem primitven Datentypen bei der Übergabe an eine Methode. Was bedeutet in diesem Zusammenhang call-by-value?

12 – Aufgabensammlung

(1) Schaffen Sie es analog zur **`DoubleListe`** eine Klasse **`IntListe`** zu entwerfen, die eine Liste vom Typ **`int`** verwaltet?

(2) Betrachten Sie das folgende Codefragment:

```
int[][] a = {{2,4,6,8}, {1,2,3}, {3,4,5}};
int[][] b = a;
int[][] c = (int[][]) a.clone();
c[2]      = a[1];
c[2][1]   = 6;
b[2][2]   = 7;

for (int i=0; i<a.length; i++)
   a[i][i]++;
```

Welche Werte haben die Ausdrücke: `a[1]==c[1]`, `b[2]==c[2]`, `a==c`, `b[2][2]`, `c[1][1]` und `c[2][2]` nach der Ausführung und warum?

(3) Wir haben ein Interface **`Haustier`** gegeben:

```
public interface Haustier {
   public String getName();
   public int getAlter();
   public String getBezeichnung();
   public String getTierstimme();
}
```

12 – Aufgabensammlung

Eine Klasse **Hund** implementiert die Methoden:

```
public class Hund implements Haustier {
   private String name;
   private int alter;

   public Hund(String name, int alter) {
      this.name  = name;
      this.alter = alter;
   }

   public String getName() {
      return name;
   }

   public int getAlter() {
      return alter;
   }

   public String getBezeichnung() {
      return "Hund";
   }

   public String getTierstimme() {
      return "wuff";
   }
}
```

Ein **`Haustierhalter`** kennt nur das Interface **`Haustier`**:

```
public class Haustierhalter {
   private Haustier meinHaustier;

   public Haustierhalter() {
      meinHaustier = null;
   }

   public void neuesHaustier(Haustier haustier) {
      meinHaustier = haustier;
   }

   public String getHaustierbezeichnung() {
      return meinHaustier.getBezeichnung();
   }
}
```

Ein kleines Testprogramm soll den Zusammenhang beider Klassen zeigen:

```
public class HaustierHaushalt {
   public static void main(String[] args) {
      Haustierhalter heinz = new Haustierhalter();
      Hund rambo = new Hund("Rambo", 3);
      heinz.neuesHaustier(rambo);
      System.out.println("Haustier von Heinz: "+
         heinz.getHaustierbezeichnung());
   }
}
```

12 – Aufgabensammlung

Machen Sie sich mit den Klassen vertraut und erweitern Sie dieses Projekt um die folgende Funktionalität:

a) Ein Haustierhalter darf mehrere Haustiere halten. Ein neues `Haustier` soll dabei über die Methode `neuesHaustier(Haustier h)` hinzugefügt werden können.

b) Erweitern Sie die Menge der Haustiere um drei Ihrer Lieblingshaustiere.

c) Haustiere leben leider nicht ewig, mal abgesehen von Tamagotchis. Erweitern Sie den Haustierhalter in der Art, dass er Haustiere auch wieder abgeben kann.