Softwareentwicklung 2 **Klausur**
Prof. Dr. H. Mössenböck **3. Juli 2003**

1. Klassen (20 Pkte)

Implementieren Sie eine Klasse *Buffer*, die ähnlich wie die Java-Klasse *StringBuffer* Zeichenketten als Character-Arrays verwaltet. Im Gegensatz zu *StringBuffer* soll in Buffer aber die maximale Länge der Zeichenketten begrenzt sein. Verwenden Sie zur Implementierung NICHT die Klasse *StringBuffer*.

```
public class Buffer {
      protected char [] buf;
```

// weitere Deklaration von Feldern 2 Punkte

// erzeugt ein neues Buffer-Objekt mit Platz für maximal size Zeichen.

```
      public Buffer(int size) {
```
// 3 Punkte

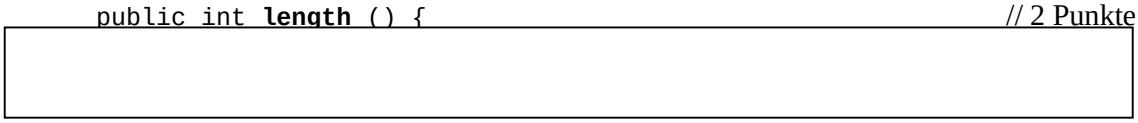
```
      }
```

// Hängt s an den Buffer an. Wenn nicht alle Zeichen von s Platz haben, sollen die überschüssigen Zeichen von s ignoriert werden.

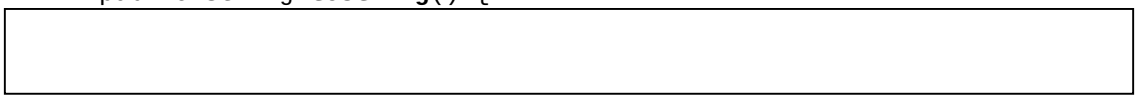
```
      public void append (String s) {
```
// 10 Punkte

```
      }
```

```
// gibt die Anzahl der Zeichen im Buffer zurück.
      public int length () {                                          // 2 Punkte
```



```
      }

// liefert den Inhalt von Buffer als String.
      public String toString() {
```



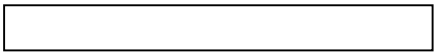
```
      }
}
```

Hinweis: Die ersten n Zeichen eines Character-Arrays *a* können mittels *new String(a,0, n)* in einen String umgewandelt werden.

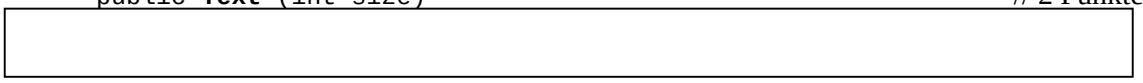
2. Vererbung (15 Punkte)

Leiten Sie aus der Klasse *Buffer* aus Aufgabe 1 eine Klasse Text ab, die es erlaubt, beliebig lange Zeichenketten zu verwalten. Zu diesem Zweck soll die Methode *append()* so überschreiben werden, dass sie den Buffer-Inhalt bei Bedarf in ein doppelt so großes charArray umkopiert und dieses zum neuen *Buffer*-Inhalt macht.
Verwenden Sie die geerbte Funktionaliät mittels Super-Calls.

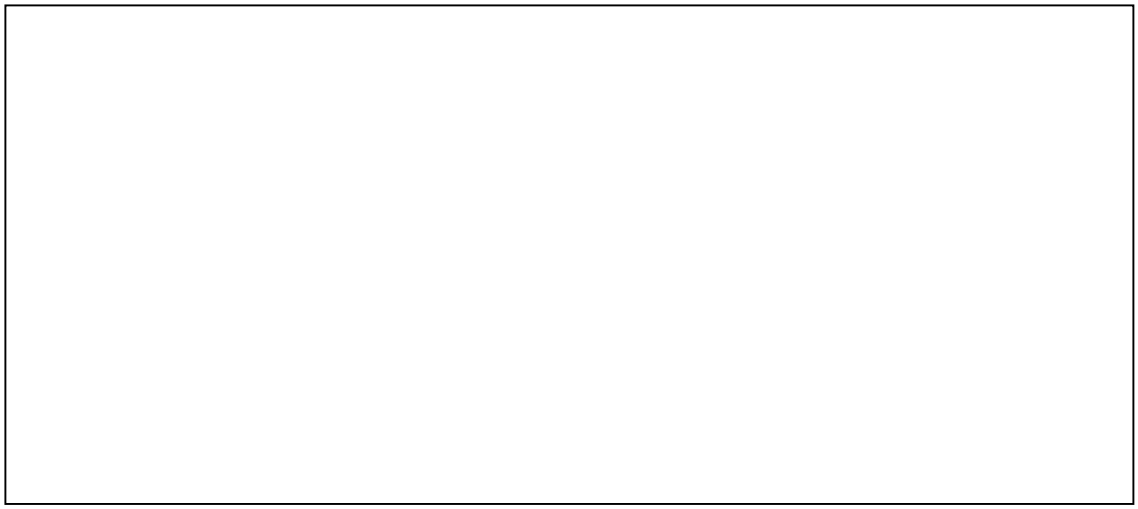
```
public class Text
```



```
{                                                                     // 1 Punkt

// Erzeugt ein neues Text-Objekt mit Platz für maximal size Zeichen.
      public Text (int size)                                          // 2 Punkte
```



```
      }

// Hängt s an den Text an. Falls zu wenig Platz dafür ist, wird die maximale Textlänge
verdoppelt.
      public void append (String s) {                                 // 12 Punkte
```

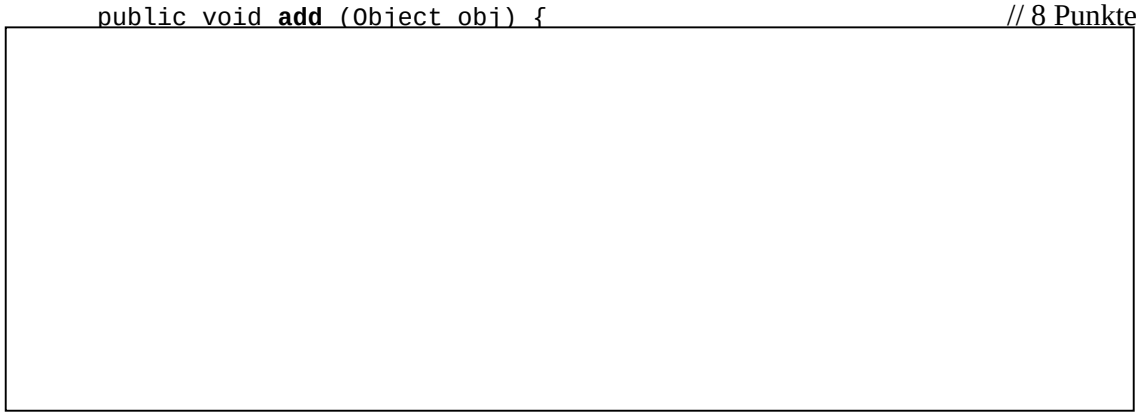


```
      }
}
```

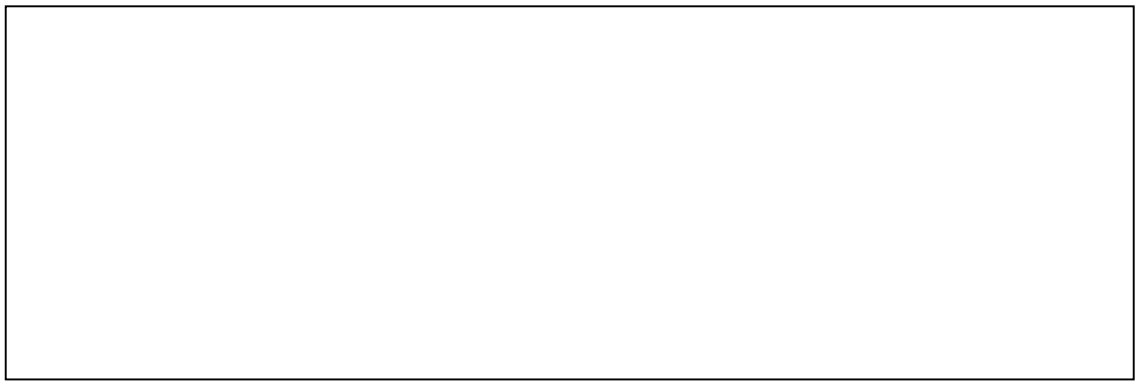
3. Frameworks (33 Punkte)

Folgende Klassen sollen eine verkettete Liste von Knoten verwalten. Implementieren Sie die Methoden *add()* und *collect()* gemäß der in den Kommentaren angegebenen Spezifikation.

```
class Node {
      Object obj;
      Node next;
      Node (Object obj) {this.obj = obj;}
}

public class List {
      Node head = null;
      Node tail = null;

// hängt obj an das Ende der Liste an
      public void add (Object obj) {                                    // 8 Punkte
```



```
      }

// liefert eine neue Liste, die jene Objekte x der ursprünglichen Liste enthält, für welche
test.qualifies(x) den Wert true ergibt.

      public List collect (Tester test) {                               // 10 Punkte
```



```
      }
}

public interface Tester {
      boolean qualifies (Object obj);
}
```

Gegeben sei nun eine Liste von Personen des folgenden Typs:

```
class Person {
      String name;
      String street;
      String city;
      int zipCode;
}
```

Diese Liste wird von einem Hauptprogramm verwendet, das eine Methode *personsFrom(city)* besitzt, die eine Liste aller Personen liefert, welche aus der Stadt *city* sind:

```
      class MainProgram {
            List list;
            …

            List personsFrom (String city) {
                  return list.collect(new CityTester(city));
            }
            ...
      }
```

Implementieren Sie die dazu nötige Klasse *CityTester*.

// 15 Punkte

4. UML-Diagramme (10 Punkte)

Zeichnen Sie ein UML-Klassendiagramm aller Typen aus Aufgabe 3. Umranden Sie dabei jenen Teil der Typen, der das Framework ausmacht.

5. Entwurfsmuster (6 Punkte)

Worin unterscheiden sich die Entwurfsmuster *Dekorator* und *Wrapper* voneinander?

6. Fragen (6 Punkte)

Was ist der Unterschied zwischen einem Interface und einer abstrakten Klasse?

Was ist der Unterschied zwischen dem statischen und dynamischen Typ einer Variablen? Wofür werden die beiden Arten von Typen verwendet?