Hinweise für den Prüfling

Auswahlzeit: 30 Minuten

Bearbeitungszeit (insgesamt): 180 Minuten

Auswahlverfahren

Es gibt zwei Aufgabengruppen A und B, aus denen jeweils ein Vorschlag zu bearbeiten ist. Der vorliegende Vorschlag aus der Gruppe A (objektorientierte Modellierung) ist ein Pflichtvorschlag.

Wählen Sie von den zwei vorliegenden Vorschlägen der Gruppe B (Konzepte und Anwendungen der theoretischen Informatik) einen zur Bearbeitung aus. Der nicht ausgewählte Vorschlag muss am Ende der Auswahlzeit der Aufsicht führenden Lehrkraft zurückgegeben werden.

Erlaubte Hilfsmittel

1. ein Wörterbuch der deutschen Rechtschreibung
2. eine Liste der fachspezifischen Operatoren

Sonstige Hinweise

ohne PC-Nutzung

In jedem Fall vom Prüfling auszufüllen

Name: _____ Vorname: _____

Prüferin/Prüfer: _____ Datum: _____

Fotoschau

Für Smartphones und Tablets gibt es Programme, die eine Fotoschau ermöglichen. Dabei werden die Fotos nacheinander für eine gewisse Zeitdauer angezeigt. In dieser Aufgabe werden nur Grundfunktionen einer solchen Fotoschau betrachtet. Der zeitliche Ablauf wird durch den Schalter „nächstes Bild“ simuliert. Zu jedem Bild werden der Titel und das Aufnahmedatum mit Uhrzeit gespeichert und angezeigt. Für einen ersten Prototypen wird der in Material 1 gegebene Konstruktor verwendet.

Aufgaben

1. In Material 2 sind die beiden Klassen *Fotoschau* und *Bild* in UML-Darstellung gegeben.

1.1 Beschreiben Sie die Klasse *Bild* ausführlich.

(8 BE)

1.2 Stellen Sie, ohne Eintragung der Attribute und Methoden, die beiden Klassen *Fotoschau* und *Bild* sowie die Klasse *JFrame* und die davon abgeleitete Klasse *GUI* (engl. Graphical User Interface) und die Klassenbeziehungen in einem UML-Klassendiagramm dar und erläutern Sie die Beziehungen.

(8 BE)

2. Die obenstehende Abbildung zeigt die Benutzeroberfläche des Programms Fotoschau.
Geben Sie die verwendeten GUI-Komponenten an und beschreiben Sie, welchem Zweck sie im Beispiel dienen.

Hinweis:
Ein Bild wird mit folgender Anweisung angezeigt:
lblFoto.setIcon(dieFotoschau.liesBildIcon()); // lblFoto ist vom Typ JLabel

(8 BE)

3. Zum Anzeigen des nächsten Fotos ruft die Klasse *GUI* zuerst die Methode *nächstesBild()* der Klasse *Fotoschau* auf und liest dann die benötigten Daten mit *liesInfo()* bzw. *liesBildIcon()*. In der Methode *nächstesBild()* wird nur das Attribut *Position* aktualisiert, in dem die Position des aktuellen Bildes gespeichert ist. Wenn „normal“ gewählt wurde, wird *Position* um 1 erhöht und nach dem letzten Bild wieder *Position* auf 0 gesetzt. In der Einstellung „zufällig“ erhält *Position* einen zufälligen, möglichen Wert. Es darf aber nicht nacheinander das gleiche Bild angezeigt werden.
Implementieren Sie die Methode *nächstesBild()*.

Verwenden Sie dazu die Programmierhilfe in Material 3.

(12 BE)

4. Die Bilder sollen nach Datum und Uhrzeit in aufsteigender Reihenfolge sortiert werden.

4.1 Erläutern Sie, warum man mit dem Format TT.MM.JJJJ hh:mm:ss von Datum und Uhrzeit nicht wie angegeben sortieren kann und geben Sie ein zum Sortieren geeignetes Format an.

(4 BE)

4.2 Implementieren Sie die Methode *liesDatumUhrzeitFormat()* der Klasse *Bild*, welche aus dem Format TT.MM.JJJJ hh:mm:ss das geeignete Format erzeugt.

Verwenden Sie dazu die Programmierhilfe in Material 3.

(7 BE)

4.3 In Material 4 ist die Implementierung der Methode *sortieren()* gegeben.
Analysieren Sie diese Methode und beschreiben Sie, wie die Bildobjekte sortiert werden.

(13 BE)

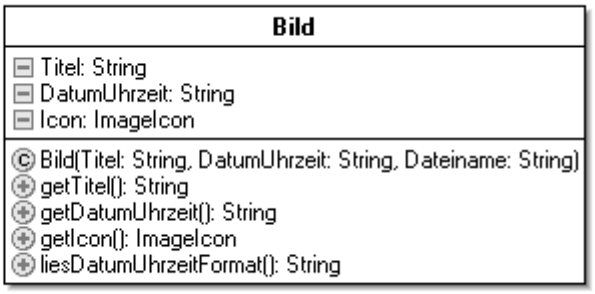
Material 1

Der Konstruktor von *Fotoschau*

```
public Fotoschau() {
  dieBilder[0] = new Bild("Segelyacht", "06.12.2013 14:56:13", "Bavaria.jpg");
  dieBilder[1] = new Bild("Dampflok BR 01", "07.08.2013 14:56:13", "BR01.jpg");
  ...
  dieBilder[5] = new Bild("Modellbahn", "01.05.1992 14:56:16", "Modellbahn.jpg");
  AnzahlBilder = 6;
  normal = true;
  Position = 0;
}
```

Material 2

Die Klassen *Fotoschau* und *Bild*



Material 3

Programmierhilfen

`int Zufallszahl = (int) (Math.random() * Anzahl)`
erzeugt eine Zufallszahl mit 0 <= Zufallszahl < Anzahl.

`public String substring(int startIndex, int endIndex)`
liefert einen Teilstring, der bei `startIndex` beginnt und bei `endIndex-1` endet.
`"hamburger".substring(4, 8)` liefert `"urge"`.

`String1.compareTo(String2)`
vergleicht zwei Strings. Steht `String1` alphabetisch vor `String2`, ist das Ergebnis negativ, bei Gleichheit 0, ansonsten positiv.

Material 4

Die Methode *sortieren()*

```
01  public void sortieren() {
02    for (int i = 1; i < AnzahlBilder; i++) {
03      int j = i;
04      Bild einBild = dieBilder[i];
05      String Datenformat = einBild.liesDatumUhrzeitFormat();
06      while (j > 0 &&
07             dieBilder[j-1].liesDatumUhrzeitFormat().compareTo(Datenformat) > 0) {
08        dieBilder[j] = dieBilder[j-1];
09        j--;
10      }
11      dieBilder[j] = einBild;
12    }
13  }
```