Lernstandserhebung 2

im Fach Programmieren – II

Fachliche Inhalte:
Programmiersprachliche Möglichkeiten
bei der Modellierung und Programmierung von Java-Klassen,
Modifizierer, abstrakte Klassen, Interfaces, Exceptions
Anwendung von ArrayLists und HashMaps

Das vorliegende Programmbeispiel modelliert die Verwaltung einer Sammlung von Fotografien.

Lesen Sie den Code und verstehen Sie seine Funktion. Nutzen Sie dazu auch das vorliegende UML-Klassendiagramm.

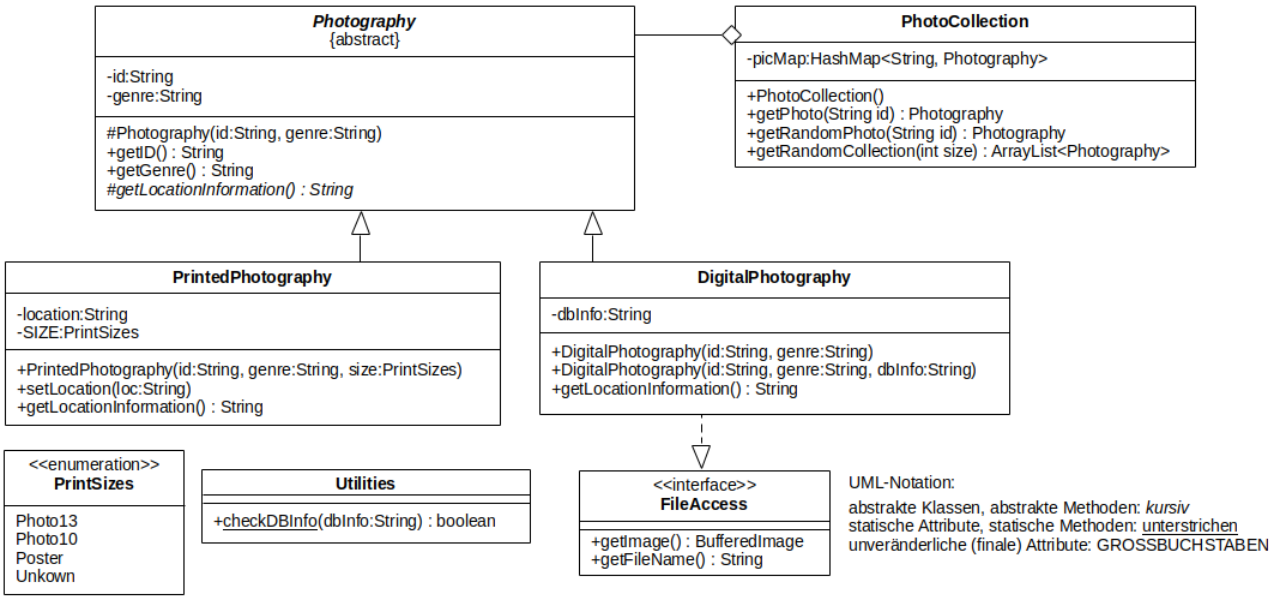
```java
package collections.photography;

public abstract class Photography {

    private String id;
    private String genre;

    protected Photography(String id, String genre) {
        this.id = id;
        this.genre = genre;
    }

    public String getID() {
        return this.id;
    }

    public String getGenre() {
        return this.genre;
    }
    public abstract String getLocationInformation();

}
```

```java
package collections.photography;

public class PrintedPhotography extends Photography{

    private final PrintSizes SIZE;
    private String location = "";

    public PrintedPhotography(String id, String genre,
            PrintSizes size) {
        super(id, genre);
        this.SIZE = size;
    }

    public void setLocation(String loc) {
        this.location = loc;
    }

    public void setSize(PrintSizes size) {
        this.size = size;
    }

    @Override
    public String getLocationInformation() {
        return this.location;
    }
}
```

```java
package collections.photography;

import java.awt.image.BufferedImage;

public class DigitalPhotography extends Photography implements FileAccess {

    // a data base connection string e.g. "Server=localhost;Database=BigPhotoDataBase;"
    private String dbInfo;

    public DigitalPhotography(String id, String genre) {
        super(id, genre);
        this.dbInfo = "";
    }

    public DigitalPhotography(String id, String genre, String dbInfo)
            throws IllegalArgumentException {
        super(id, genre);
        if (Utilities.checkDBInfo(dbInfo) == false)
            throw new IllegalArgumentException("Invalid DB Information");
        this.dbInfo = dbInfo;
    }

    @Override
    public String getLocationInformation() {
        return this.dbInfo;
    }
}
```

```java
package collections.photography;

public enum PrintSizes {
    Photo13,
    Photo10,
    Poster,
    Unkown
}
```

```java
package collections.photography;

import java.util.ArrayList;
import java.util.HashMap;

public class PhotoCollection {

    private HashMap<String, Photography> picMap = new HashMap<String, Photography>();

    public PhotoCollection() { }

    public void addPhoto(Photography photo) throws PhotoAlreadyExistsException {
        String id = photo.getID();
        if (picMap.get(id) != null)
            throw new PhotoAlreadyExistsException(id);
        picMap.put(id, photo);
    }

    public Photography getPhoto(String id) {
        return picMap.get(id);
    }

    public Photography getRandomPhoto() {
        Object[] photos =  (Object[]) picMap.values().toArray();
        int n = photos.length;
        int idx = (int)(n*Math.random());
        Photography photo = (Photography) photos[idx];
        return photo;
    }

    public ArrayList<Photography> getRandomCollection(int size) {..
    }
}
```



Hinweis: Der Begriff „Coderumpf“ umfasst den vollständigen Methodennamen, ggf. eine zutreffende Annotation und einen passenden Default-Rückgabewert.
Beispiel:

```
@Annotation
private static DataType doSomething(DataType var) {
        // TODO Auto-generated method stub
        return propperDefaultValue;
}
```

Aufgabe 1
Vervollständigen Sie die Implementierung der Klasse DigitalPhotography, indem Sie die Coderümpfe der zwei fehlenden Methoden angeben.

Bewertung: PFItf1: …. / 4 Pkt. PFItf2: …. / 4 Pkt.

Aufgabe 2
Die Methode checkDBInfo wird im Konstruktor der Klasse DigitalPhotography dazu genutzt, um sicherzustellen, dass ein korrekter Datenbank-Connect-String in der Variable dbInfo vorliegt. Geben Sie den Coderumpf der Methode „checkDBInfo“ an. Ergänzen Sie ebenfalls den Namen der Klasse, in der die Methode implementiert wird!

```
package collections.photography;

public class ...                                        {




}
```

Bewertung: PFCln: …. / 1 Pkt. PFMod: …. / 3 Pkt. PFMeth: …. / 2 Pkt. PFRet: …. / 1 Pkt.

Aufgabe 3
Schreiben Sie einen JavaDoc-Kommentar für die vorliegende Implementierung des Konstruktors
`DigitalPhotography(String id, String genre, String dbInfo)`

Bewertung: PFInfo: …. / 1 Pkt. PFPar: …. / 3 Pkt. PFThr: …. / 1 Pkt.

Aufgabe 4
Geben Sie die Implementierung der PhotoAlreadyExistsException – Klasse an!

Bewertung: CUEx: …. / 3 Pkt. CUPb: …. / 2 Pkt. CUSp: …. / 1 Pkt.

Aufgabe 5
Programmieren Sie die Methode getRandomCollection der Klasse PhotoCollection!
Zufällige Mehrfachauswahl von Fotos ist dabei zulässig.

Bewertung: PSCl: …. / 3 Pkt. PSLp: …. / 2 Pkt. PSMt: …. / 2 Pkt.
PSAd: …. / 3 Pkt. PSIt: …. / 1 Pkt. PSRt: …. / 2 Pkt.

Aufgabe 6
Was bedeutet die Annotation @Override?

Bewertung: CUOvr: …. / 3 Pkt.

Aufgabe 7
Gegeben ist der folgende Java-Code.

```
package collections.photography;

public class PhotoDemo {

	public static void main(String[] args)
	{
		PrintedPhotography p1 = new PrintedPhotography("p1Haus","architecture",
                                                                PrintSizes.Photo10);
		DigitalPhotography d1 = new DigitalPhotography("img1234","nature");
		DigitalPhotography d2 = new DigitalPhotography("img567","trees");
		DigitalPhotography d3 = new DigitalPhotography("img999","people");

		PhotoCollection pc = new PhotoCollection();

		pc.addPhoto(d1);
		pc.addPhoto(p1);
		pc.addPhoto(d2);
		pc.addPhoto(d3);

		Photography p = pc.getPhoto("img1234");
		System.out.println(p);

		p = pc.getRandomPhoto();
		System.out.println(p);
	}
}
```

Implementieren Sie die notwendige Ausnahmebehandlung!

Bewertung: PFEx: …. / 4 Pkt.

Aufgabe 8
Beschreiben Sie die Klasse PhotoCollection und ihre Implementierung verbal in kurzen aussagekräftigen Sätzen!

Bewertung: PFSt: …. / 1 Pkt.

Aufgabe 9
Vervollständigen Sie die folgenden Aussagen an den gekennzeichneten Stellen!

Mit Hilfe von ..... wird angegeben, dass eine Klasse vorgegebene Funktionalität/Verhalten bereit stellt.

..... dienen dazu, eine Rahmenstruktur von Objekten vorzugeben.

Als „final“ gekennzeichnete Klassen .....

Eine mit „static“ gekennzeichnete Methode ……..

Bewertung: CUItf: …. / 1 Pkt. CUAbC: …. / 1 Pkt. CUFin: …. / 1 Pkt. CUKM: …. / 1 Pkt.

Erreichte Gesamtpunktzahl: ……. / 55 Pkt.