Humboldt-Universität zu Berlin
Grundlagen der Programmierung
(Vorlesung von Prof. Bothe)

Institut für Informatik
WS 15/16

Übungsblatt 6: Polymorphie und Generics

Abgabe: bis **9:00 Uhr** am **29.01.2016** über **Goya**

Die Lösung dieses Übungsblatts soll nach Möglichkeit in Gruppen von je 2 Personen erfolgen. Die Abgabe der Lösungen erfolgt durch Hochladen **einer Datei für jede Aufgabe** im Goya-System. Verwenden Sie exakt den in der Aufgabe angegebenen Dateinamen! **Java-Lösungen** müssen im UTF-8 Format abgegeben werden und auf dem Institutsrechner **`gruenau5`** korrekt kompilierbar sein (ansonsten wird die Aufgabe mit 0 Punkten bewertet).

Aufgabe 1 (Variablen- und Methodenbindung → `Blatt06_Aufgabe1.txt`) 6 Punkte

Vollziehen Sie folgenden Java-Code nach. Überlegen Sie, welche Ausgaben ein Aufruf der PolyTest-Main-Methode generiert und warum. Nehmen Sie dabei an, dass jede Zeile der Main-Methode ausgeführt wird, auch wenn zuvor ein Fehler aufgetreten sein sollte. Weiterhin darf angenommen werden, dass alle Klassen dem selben Paket angehören.

```java
class Base {
    protected String s = "base";
    protected String t = "base2";
    public String s() { return s; }
    public String t() { return t; }
}
```

```java
class Derived extends Base {
    protected String s = "derived";
    protected String t = "derived2";
    public Derived() { }
    public String s() { return s; }
}
```

```java
public class PolyTest {
    static void out(Object o) {
        System.out.println(o);
    }
    public static void main(String[] args) {
        Derived d = new Derived();
        out( d.s );
        Base b = d;
        out( b.s );
        out( d.s() );
        out( b.s() );
        out( d.t() );
        out( b.t() );
    }
}
```

Wählen Sie aus folgenden Multiple-Choice-Fragen zu Zeile 7 bis 13 die zutreffenden Antworten aus und speichern Sie das Ergebnis im weiter unten angegebenen Format in der Datei `Blatt06_Aufgabe1.txt`. Es können pro Frage mehrere Antworten richtig sein. Jede richtige Antwort entspricht einem halben Punkt, jede falsche Antwort entspricht einem halben Minuspunkt. Die Beantwortung einer Frage kann jedoch nicht negativ werden.

Zeile 7:

a) Ein Zugriff auf eine protected-Variable ist aus diesem Kontext nicht zulässig.
b) Der Zugriff auf die String-Variable geschieht durch statische Bindung.
c) Der Zugriff auf die String-Variable geschieht durch dynamische Bindung.
d) Die Konsolen-Ausgabe dieser Zeile ist „base“.
e) Die Konsolen-Ausgabe dieser Zeile ist „derived“.

Zeile 9:

a) Ein Zugriff auf eine protected-Variable ist aus diesem Kontext nicht zulässig.
b) Der Zugriff auf die String-Variable geschieht durch statische Bindung.
c) Der Zugriff auf die String-Variable geschieht durch dynamische Bindung.
d) Die Konsolen-Ausgabe dieser Zeile ist „base“.
e) Die Konsolen-Ausgabe dieser Zeile ist „derived“.

Zeile 10:

a) Der Zugriff auf die s()-Methode geschieht ausschließlich durch statische Bindung.
b) Der Zugriff auf die s()-Methode geschieht durch dynamische Bindung.
c) Die Konsolen-Ausgabe dieser Zeile ist „base“.
d) Die Konsolen-Ausgabe dieser Zeile ist „derived“.

Zeile 11:

a) Der Zugriff auf die s()-Methode geschieht ausschließlich durch statische Bindung.
b) Der Zugriff auf die s()-Methode geschieht durch dynamische Bindung.
c) Die Konsolen-Ausgabe dieser Zeile ist „base“.
d) Die Konsolen-Ausgabe dieser Zeile ist „derived“.

Zeile 12:

a) Der Zugriff auf die t()-Methode geschieht ausschließlich durch statische Bindung.
b) Der Zugriff auf die t()-Methode geschieht durch dynamische Bindung.
c) Die Konsolen-Ausgabe dieser Zeile ist „base2“.
d) Die Konsolen-Ausgabe dieser Zeile ist „derived2“.

Zeile 13:

a) Der Zugriff auf die t()-Methode geschieht ausschließlich durch statische Bindung.
b) Der Zugriff auf die t()-Methode geschieht durch dynamische Bindung.
c) Die Konsolen-Ausgabe dieser Zeile ist „base2“.
d) Die Konsolen-Ausgabe dieser Zeile ist „derived2“.

Abgabeformat (Beispiel):

```
7: a,d
9: b
...
```

Aufgabe 2 (Überladung und Polymorphismus → `Poly[1,2].java`) **8 Punkte**

Betrachten Sie die folgenden Klassen `Poly1`, `Poly2` und `Poly`.

```java
public class Poly1 {
    public void f() {
        System.out.println( "1f" );
        g();
    }
    private void g() {
        System.out.println( "1g" );
        h( 10 );
    }
    protected void h( int i ) { System.out.println( "1hi" ); }
    void h( byte b ) { System.out.println( "1hb" ); }
}
```

```java
public class Poly2 extends Poly1 {
    protected void f() {
        System.out.println( "2f" );
        g();
        h( 12 );
    }
    void g() {
        System.out.println( "2g" );
        h( 18 );
    }
    public void h( int i ) { System.out.println( "2hi" ); }
    public void h( byte b ) { System.out.println( "2hb" ); }
}
```

```java
public class Poly {
    public static void main( String args[] ) {
        Poly1 a = new Poly1();
        a.g();
        Poly2 b = new Poly2();
        b.f();
    }
}
```

Verändern Sie die Modifier und die Rümpfe der Methoden, sodass das Programm die folgende Ausgabe erzeugt: Dabei dürfen Sie keine Befehle hinzufügen, verändern, oder entfernen, die eine Ausgabe auf der Konsole erzeugen.

```
1g
1hb
2f
1g
2hb
1hb
```

Aufgabe 3 (Interfaces und Generics → `Aufgabe3.zip`) **6 Punkte**

Folgende Zusammenhänge wurden modelliert:

- Es gibt Verbindungen (`AbstractConnection`) in ihren speziellen Ausprägungen der Telefonverbindung (`Phone`) und der Mail (`Mail`).
- Weiterhin gibt es Personen (`Person`).
- Sowohl Personen, als auch Verbindungen verpflichten sich, die Methode `void call()` zu implementieren, indem Sie die Schnittstelle `Callable` implementieren.
- Eine Verbindung fordert außerdem die Methode `void connect()`, die jedoch erst in ihren Spezialisierungen implementiert wird.
- Eine Verbindung hat stets eine (Ziel-)Person.
- Der `AutoDialer` ist eine generische Klasse, die eine Liste von Elementen speichert, welche die Schnittstelle `Callable` implementieren.
- Einem AutoDialer können Elemente hinzugefügt werden (`insert`) und es können alle Elemente „aufgerufen“ werden (`callAll`).
- Es ist möglich, Instanzen von `AutoDialer` zu erstellen, die entweder nur Personen, oder nur Verbindungen jedoch nicht beides zugleich speichern können.

In der Datei `Aufgabe3.zip` sind die entsprechenden Quelltexte hinterlegt. Leider sind (vielleicht aufgrund eines Fehlers im Komprimierungsalgorithmus) die jeweils erste Zeilen der Quelltextdateien verloren gegangen. Fügen Sie jeweils nur die erste Quelltextzeile ein, ohne den bestehenden Code zu verändern.