ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Schriftlicher Test über den Stoff der Vorlesung
Programmieren 1 für Wirtschaftsinformatiker

Name: _____ Vorname: _____

Matr.Nr.: _____ Studiengang: _____

Semester: WS 14/15

Ich bestätige, dass ich die folgenden Anmerkungen gelesen und mich von der Vollständigkeit dieses Klausurexemplares (Seiten 1 – 23) überzeugt habe.

_____
Unterschrift des o.g. Klausurteilnehmers
bzw. der o.g. Klausurteilnehmerin

Anmerkungen:

1. Legen Sie bitte Ihren Studierendenausweis bereit.
2. Bitte tragen Sie *Name*, *Vorname, Studiengang* und *Matrikelnummer* deutlich lesbar in den Feldern oben ein.
3. Dieser Test beinhaltet **12 Aufgaben** mit insgesamt **110 Punkten**. Die erreichte Portfoliopunkteanzahl entspricht dem Minimum aus Ihrer erreichten Punktzahl und 70.
4. Folgende Hilfsmittel sind zugelassen: **keine.**
5. Täuschungsversuche führen zu Nichtbestehen der Modulprüfung.
6. Unleserliche oder mit Bleistift geschriebene Lösungen werden nicht gewertet.
7. Die Bearbeitungszeit beträgt **70 Minuten**. Es gibt 10 Minuten Einlesezeit.
8. Die Aufgaben sind entweder auf Englisch oder auf Deutsch, aber pro Aufgabe durchgehend in genau einer Sprache, zu beantworten.

Nur für den Prüfer: (maximal 70 von 110 Punkten)

1:6	**2**:5	**3**:6	**4**:10	**5**:10	**6**:14	**7**:8	**8**:12	**9**:10	**10**:7	**11**:10	**12**:12	Σ

Aufgabe 1 – Programmierfehler **(2 + 2 + 2 = 6 Punkte)**

Die folgenden Sourcecodeauszüge beinhalten Fehler, die entweder dafür sorgen, dass das Programm nicht kompiliert werden kann, oder die erst zur Laufzeit auftreten. Geben Sie für jeden Codeauszug jeweils an, ob es sich um einen Compilerfehler oder einen Laufzeitfehler handelt. Benennen Sie auch jeweils kurz den Fehler – das kann einerseits der Name einer Exception oder eine ähnlich kurze (aber eindeutige) Information sein. Pro richtig analysierten Codeabschnitt erhalten Sie 2 Punkte.

```
// Aufgabenteil A
int[] a;
a[0] = 1;
System.out.println(a[-1] + a[0] + a[1]);
```

O Compilerfehler O Laufzeitfehler

Ursache:

_____

```
// Aufgabenteil B
int i = 3;
if (i > 2) {
	int a = 7;
} else {
	int a = 5;
}
System.out.println(a);
```

O Compilerfehler O Laufzeitfehler

Ursache:

_____

```
// Aufgabenteil C
int i =0;
while(i<5){
	System.out.println("1/i="+(1/i++));
	i -= 1;
}
```

O Compilerfehler O Laufzeitfehler

Ursache:

Aufgabe 2 – Methodenüberladung **(5 Punkte)**

Gegeben ist unten stehendes Programm. Kreuzen Sie unten an, was die Aufrufe der überladenen Methode bar() jeweils ausgeben.

```
public class Foo {
	static void bar(int a, double b) {
		System.out.print("ID ");
	}
	static void bar(int a, int b) {
		System.out.print("II ");
	}
	static void bar(double a, long b) {
		System.out.print("DL ");
	}
	static void bar(float... a) {
		System.out.print("FS ");
	}
	static void bar(int a, byte b){
		System.out.print("IB ");
	}
	static void bar(double a, double b){
		System.out.print("DD ");
	}
	public static void main(String[] args) {
		int i=0;
		long l =0;
		float f=0;
		double d=0;
		char c='0';
		bar(i,f);
		bar(d,i);
		bar(i,c,i);
		bar(i,c);
		bar(l,c);
	}
}
```

bar(i,f);	*bar*(d,i);	*bar*(i,c,i);	*bar*(i,c);	*bar*(l,c);
O ID	O ID	O ID	O ID	O ID
O II	O II	O II	O II	O II
O DL	O DL	O DL	O DL	O DL
O FS	O FS	O FS	O FS	O FS
O IB	O IB	O IB	O IB	O IB
O DD	O DD	O DD	O DD	O DD

Aufgabe 3 – Vererbung **(6 Punkte)**

Gegeben ist unten stehendes Programm. Kreuzen Sie unten an, was das Programm beim Start der Klasse Vererbung ausgibt.

```
public class Vererbung {
	public static void main(String[] args) {
		Apfel a = new Apfel();
	}
}

class Apfel extends Frucht {
	static Print mp = new Print("Boskop!");
	Print mp2 = new Print("Gala!");

	public Apfel() {
		System.out.print("Jonagold!");
	}
}

class Frucht extends Essbar {
	Print mp2 = new Print("Banane!");

	public Frucht() {
		System.out.print("Kirsche!");
	}
}

class Essbar {
	static Print mp = new Print("Essbar!");
}

class Print {
	public Print(String msg) {
		System.out.print(msg);
	}
}
```

O Banane!Kirsche!Gala!Jonagold!

O Essbar!Boskop!Banane!Kirsche!Gala!Jonagold!

O Banane!Kirsche!Gala!Jonagold!Essbar!Boskop!

O Boskop!Essbar!Gala!Banane!Jonagold!Kirsche!

O Essbar!Boskop!Kirsche!Banane!Jonagold!Gala!

O Boskop!Essbar!Gala!Jonagold!Banane!Kirsche!

Aufgabe 4 – Objekte **(10 Punkte)**

Gegeben ist unten stehendes Programm. Geben Sie in der Tabelle unten für die jeweilig genannte Variable den Wert an, den diese jeweils an der mit `//hier` markierten Stelle hat. Für jede richtige Zeile erhalten Sie einen Punkt. Tipp: Skizzieren Sie, welche Objekte wie erstellt und verändert werden.

```
public class Hausbauer {
	static Baumhaus bauHaus(int hoehe, int breite) {
		Baumhaus b = new Baumhaus();
		b.hoehe = hoehe;
		b.breite = breite;
		return b;
	}

	static Baumhaus machBreiter(Baumhaus b) {
		Baumhaus bb = new Baumhaus();
		bb.hoehe = b.hoehe;
		bb.breite = b.breite + 1;
		return bb;
	}

	static Baumhaus machHoeher(Baumhaus b) {
		b.hoehe++;
		return b;
	}

	public static void main(String[] args) {
		Baumhaus b = bauHaus(2, 3);
		Baumhaus c = machBreiter(b);
		c.nachbar = b;
		Baumhaus d = machHoeher(b);
		d.nachbar = b;
		++c.hoehe;
		Baumhaus e = machHoeher(b);
		e.nachbar = c;
		e.breite = b.breite - 1;
		c.hoehe++;
		c.breite -= 2;
		boolean bUndCBenachbart = (b.nachbar == c || c.nachbar == b);
		//hier
	}
}

class Baumhaus {
	public int hoehe;
	public int breite;
	public Baumhaus nachbar;
	public int nummer = ++naechsteNummer;
	static int naechsteNummer = 0;
}
```

Variable	Wert
bUndCBenachbart	
b.breite	
b.hoehe	
c.breite	
c.hoehe	
d.breite	
d.hoehe	
e.breite	
e.hoehe	
Baumhaus.*naechsteNummer*	

Leerseite für Notizen zu Aufgabe 4

Aufgabe 5 – Instanz- und Klassenmethoden, Zugriffsmodifikatoren (10 Punkte)

Gegeben ist unten stehendes Programm, welche der Aufrufe in der main-Methode sind zulässig? Kreuzen Sie dafür die Tabelle daneben an. Für jede richtig angekreuzte Zeile erhalten Sie 1 Punkt, für jede falsch angekreuzte Zeile einen halben Minuspunkt. Insgesamt können Sie in dieser Aufgabe nicht weniger als 0 Punkte erreichen. Tipp: Alle Klassen liegen im gleichen Package.

```
public class Reise {
	public static void main(String[] args) {
		Verkehrsmittel v = new Cessna();
		v.flieg();
		v.baujahr = 1990;
		v = new Flugzeug();
		Auto.getPreis();
		Auto b = new Auto();
		v = new Auto();
		v.bewegDich();
		b.bewegDich();
		b.getPreis();
	}
}

class Auto extends Verkehrsmittel {
	public void bewegDich() {
		System.out.println("Auto");
	}
	public int getPreis() {
		return super.getPreis();
	}
}

abstract class Flugzeug extends Verkehrsmittel {
	public static void flieg() {
		System.out.println("Hui");
	}
}

class Cessna extends Flugzeug {
}

class Verkehrsmittel {
	public String name = "";
	protected int baujahr;
	private int preis;
	protected int getPreis(){
		return this.preis;
	}
	private void bewegDich() {
		System.out.println("Bewegung");
	}
}
```

Zeile	Zulässig	Unzulässig
`Verkehrsmittel v = new Cessna();`		
`v.flieg();`		
`v.baujahr = 1990;`		
`v = new Flugzeug();`		
`Auto.getPreis();`		
`Auto b = new Auto();`		
`v = new Auto();`		
`v.bewegDich();`		
`b.bewegDich();`		
`b.getPreis();`		

Aufgabe 6 – Algorithmen und Datenstrukturen **(2 + 2 + 7 + 3 = 14 Punkte)**

a) Die Effizienz von Algorithmen wird durch die Zuordnung zu Komplexitätsklasssen beschrieben. Nennen Sie die beiden Komplexitätsklassen, in denen Standardsortieralgorithmen typischerweise liegen.

b) Gegeben ist der folgende Codeausschnitt. Um welche Art der Ihnen bekannten Algorithmen handelt es sich hierbei?

```
static Integer tuWas(int wert, ArrayList<Integer> werte) {
	Iterator<Integer> it = werte.iterator();
	while (it.hasNext()) {
		if (it.next() == wert)
			return wert;
	}
	return null;
}
```

c) Gegeben ist folgender Ausschnitt aus einer Queueimplementierung. Ergänzen Sie den fehlenden Inhalt der Methode haengeAn(T), so dass diese neue Elemente am Ende der Queue einfügt. Achten Sie darauf, die Generics korrekt zu verwenden. Alle Klammern für die Methode haengeAn(T) sind bereits korrekt gesetzt.

```
public class Warteschlange<T> {

	private Eintrag<T> anfang;

	public T entnehme() {
		if (anfang == null)
			return null;
		Eintrag<T> ergebnis = anfang;
		anfang = ergebnis.naechster;
		return ergebnis.wert;
	}

	public String toString() {
		StringBuffer res = new StringBuffer("[");
		Eintrag<T> aktuell = anfang;
		while (aktuell != null) {
			res.append(aktuell.wert + " ");
			aktuell = aktuell.naechster;
		}
		return res.toString().trim() + "]";
	}
```

```
	public void haengeAn(T wert) {
		//Hier geht's los
```

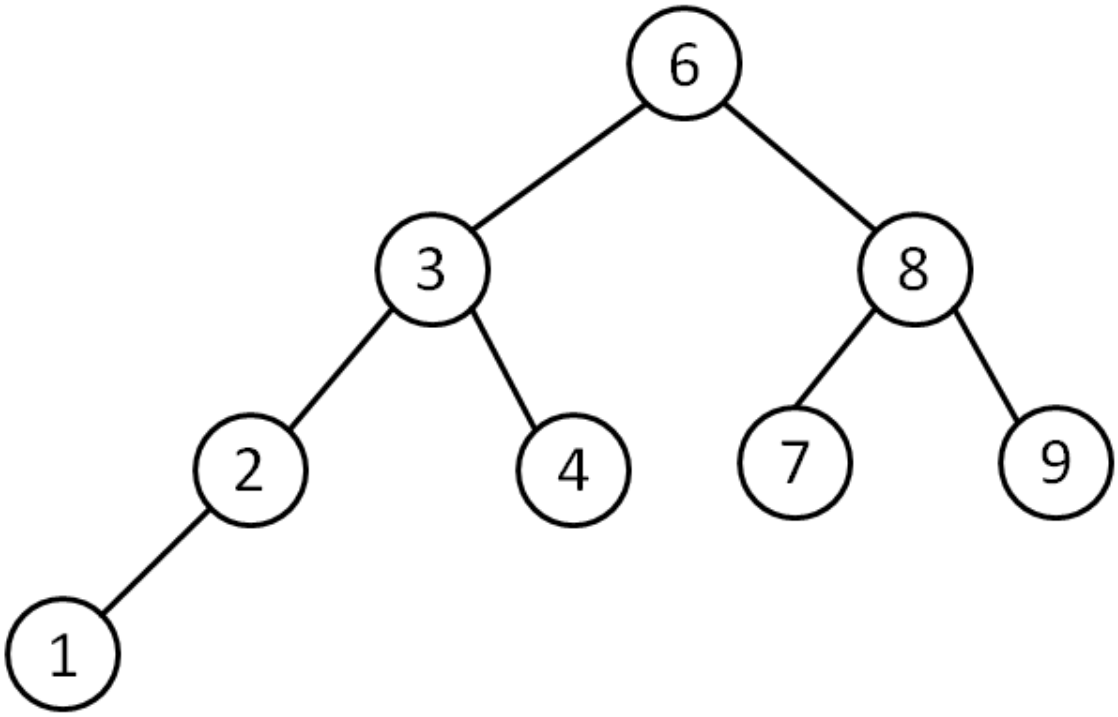
```
	}
}

class Eintrag<T> {

	T wert;
	Eintrag<T> naechster;
}
```

Leerseite, falls Sie Zusatzplatz zum Bearbeiten brauchen

d) Gegeben ist folgender sortierter Binärbaum. Notieren Sie dessen Postorderdarstellung und skizzieren Sie anschließend im Bild, an welcher Stelle der neue Wert „5“ eingefügt werden müsste. Die Postorderdarstellung soll die 5 noch ***nicht*** beinhalten.



Aufgabe 7 – Streams **(8 Punkte)**

Schreiben Sie Sourcecode, mit dem Sie den Inhalt einer Datei „file.txt“ von der Festplatte einlesen und zeilenweise in einem Objekt vom Typ ArrayList speichern. Gehen Sie dabei davon aus, dass es sich bei der Datei um eine Textdatei, d.h. keine Binärdatei, handelt. Falls IOExceptions auftreten, so sollen diese in eine neue RuntimeException verpackt und dann geworfen werden. Stellen Sie sicher, dass das Programm danach ordnungsgemäß beendet ist, ohne Systemressourcen in einem undefinierten Zustand zu hinterlassen.
Tipp: Nutzen Sie beim Einlesen die Klasse BufferedReader. Importanweisungen sowie Methoden- und Klassenrumpf sind nicht nötig. Nutzen Sie ggf. die Rückseite, falls Sie zusätzliche Platz benötigen.

Aufgabe 8 – Concurrency **(3 + 6 + 3 = 12 Punkte)**

Gegeben ist die folgende Klasse MessageQueue:

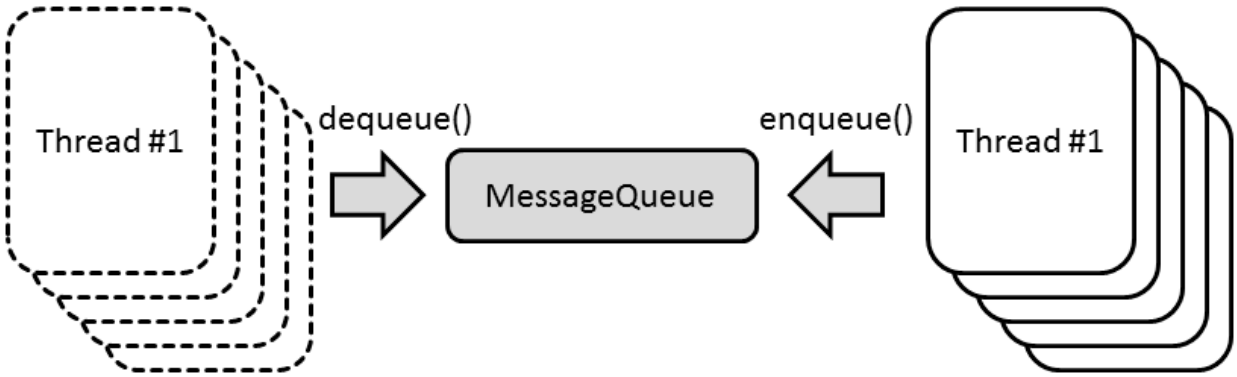
```
public class MessageQueue<T> {

	private ArrayList<T> queue = new ArrayList<>();

	public void enqueue(T message) {
		queue.add(message);
	}

	public T dequeue() {
		if (queue.size() > 0){
			return queue.remove(0);
		}
		return null;
	}
}
```

Diese Klasse soll zur Kommunikation zwischen Threads genutzt werden, d.h. eine Gruppe von Threads erstellt (wie im Bild unten skizziert) Nachrichten, während eine zweite Gruppe von Threads diese Nachrichten abruft und jeweils bearbeitet.



a) Beschreiben Sie kurz, warum der obige Sourcecode zu Problemen führen kann, wenn mehrere Threads gleichzeitig mit einem Objekt dieser Klasse interagieren. Zu welchen Problemen kann es kommen?

b) Ändern Sie den unten erneut notierten Sourcecode der Klasse MessageQueue<T> so ab, dass der Zugriff mehrerer Threads gleichzeitig nicht mehr zu Problemen führen kann. Fügen Sie dabei nur zusätzliche Codezeilen ein, das Verändern bestehender Codezeilen wird als falsch gewertet.

```
public class MessageQueue<T> {
	private ArrayList<T> queue = new ArrayList<>();


	public void enqueue(T message) {


		queue.add(message);


	}
	public T dequeue() {


		if (queue.size() > 0){


		return queue.remove(0);

		}


		return null;


	}
}
```

c) Halten Sie es für eine gute Idee, eine solche Messagequeue zur Kommunikation von Threadgruppen zu nutzen? Fallen Ihnen Argumente dafür oder dagegen ein? Begründen Sie kurz Ihre Meinung.

Aufgabe 9 – Patterns **(4 + 4 + 2 = 10 Punkte)**

Gegeben ist der untenstehende korrekte (aber nicht besonders sinnreiche) Sourcecode. Dort hat sich ein Ihnen aus der Vorlesung bekannntes OO-Designpattern eingeschlichen.

a) Benennen Sie dieses Designpattern und markieren Sie grob (bspw. durch Umkringeln) dafür relevante Stellen im Code.

b) Was bewirkt dieses Designpattern und wann verwendet man es?

c) Was sind eigentlich Design Patterns und warum sollte man darüber nachdenken, sie zu nutzen?

```
public class DonutFactory {

	public static void createDonut() {
		Donut[] donuts = new Donut[10];
		for (int i = 0; i < donuts.length; i++) {
			donuts[i] = new Donut(getFlavor());
		}
		boolean moreSugar = true;
		while (moreSugar) {
			moreSugar = false;
			for (int i = 0; i < donuts.length; i++) {
				if (donuts[i].flavor.compareTo(donuts[i + 1].flavor) > 0) {
					moreSugar = true;
					Donut yummi = donuts[i];
					donuts[i] = donuts[i + 1];
					donuts[i + 1] = yummi;
				}
			}
		}
	}

	public static String getFlavor() {
		if (Math.random() > 0.5)
			return "Chocolate";
		return "Plain";
	}
}

class Factory {

	private static Factory creator = new Factory();

	public static Factory getFactory() {
		return creator;
	}

	private Factory() {
		System.out.println("Strategy prepared.");
	}


}

class Donut {
	String flavor;

	public Donut(String flavor) {
		super();
		this.flavor = flavor;
	}

}
```

Aufgabe 10 – Diverses **(7 Punkte)**

Markieren Sie in der folgenden Tabelle, welche der Aussagen jeweils wahr bzw. falsch sind. Für richtig angekreuzte Zeilen erhalten Sie dabei einen Punkt, für falsch angekreuzte Zeilen -0,5 Punkte und für nicht angekreuzte Zeilen 0 Punkte. Insgesamt können Sie in dieser Aufgabe nicht weniger als 0 Punkte erreichen.

	Wahr	**Falsch**
`int i = 3; System.out.println(--i);` gibt 2 aus und i hat danach den Wert 2.		
Mit dem Aufruf der Instanzmethode `sleep(long)` aus der Klasse `Thread` ist es möglich, andere Threads für ein bestimmtes Zeitintervall „schlafen zu legen“.		
Die gezielte Verwendung von Gettern und Settern ermöglicht es, anderen Klassen zwar Lese- aber keinen Schreibzugriff auf Instanzvariablen zu gewähren.		
Der Aufruf `List<Number> l = new ArrayList<Integer>()` ist zulässig, da Integer Kindklasse von Number ist.		
Die Methode `void foo(List<? extends Number> list){…}`kann mit LinkedList-Instanzen aufgerufen werden, die mit Double typisiert sind.		
Die generische Klasse HashSet kann mit `byte[]` aber nicht mit `byte` typisiert werden.		
Unter Refactoring versteht man, bestehenden Sourcecode in seiner Struktur und seinem Aufbau zu verändern, ohne die Funktionalität zu beeinflussen. D.h. nur Interna werden verändert, ohne dass die Änderungen nach außen sichtbar werden.		

Aufgabe 11 – Programmierpraxis **(10 Punkte)**

Schreiben Sie eine Klasse Ausgeber, die das Interface Runnable implementiert. Erzeugen und starten Sie in der main-Methode einen neuen Thread auf Basis eines Ausgeberobjektes. Dieser Thread soll dann die Zahlen von 0 bis 50 durchgehen und alle durch 4 teilbaren Zahlen ausgeben. Die Zahl 40 soll dabei jedoch übersprungen werden.

Aufgabe 12 – Programmentwurf **(2 + 3 + 3 + 4 = 12 Punkte)**

a) Gegeben sind die folgenden Klassen Apfel, Birne, Obst und Brot sowie das Interface Essbar. Ergänzen Sie die Klassen/das Interface, so dass diese auf sinnvolle Art und Weise eine gemeinsame Vererbungshierarchie bilden.

```
public class Apfel {

}

public class Birne {

}

public class Obst {

}

public class Brot {

}

public interface Essbar {

}
```

b) Schreiben Sie nun eine Klasse Obstkorb, diese soll lediglich eine öffentliche Instanzvariable zum Speichern von Äpfeln und Birnen beinhalten. Begründen Sie kurz, warum und welchen Datentyp Sie für diese Instanzvariable gewählt haben.
Tipp: Sie benötigen weder Konstruktoren noch Getter/Setter, import-Anweisungen oder Kommentare. Notieren Sie einfach nur Deklarierung und Initialisierung der betreffenden Instanzvariable sowie die erste Zeile der Klasse („public class…“). Jede Klasse kommt inklusive Klammern mit maximal drei Zeilen aus

c) Ergänzen Sie nun eine Klasse Einkaufszettel, diese soll für jeden einzukaufenden Artikel (bspw. Obst oder Brot) jeweils die Anzahl festhalten, die eingekauft werden soll. Auch hier ist lediglich eine öffentliche Instanzvariable notwendig (Tipp aus b) gilt auch hier). Begründen Sie erneut kurz, warum und welchen Datentyp Sie gewählt haben.

d) Es soll nun auch möglich sein, Einkaufszettel für andere Produkte (bspw. Schrauben oder Nägel) zu haben. Schreiben Sie eine einfache abstrakte Klasse BasisEinkaufszettel, die an die bisherige Klasse Einkaufszettel angelehnt ist. Diese soll daher generisch sein. Schreiben Sie nun eine neue Version der Klasse Einkaufszettel und lassen Sie diese auf geeignete Art und Weise von BasisEinkaufszettel erben.

Zusätzlicher Platz zum Bearbeiten falls nötig.