1. Aufgabe

Bei einer Lotterie soll der folgende Gewinnplan gelten:

- Durch 13 teilbare Losnummern gewinnen 100 Euro.
- Losnummern, die nicht durch 13 teilbar sind, gewinnen immerhin noch einen Euro, wenn sie durch 7 teilbar sind.

Wird in folgendem Codefragment für Losnummern in der Variablen `losNr` der richtige Gewinn ermittelt?

```
if (losNr % 13 != 0)
      if (losNr % 7 == 0)
      System.out.println("Das Los gewinnt einen Euro!");
else
      System.out.println("Das Los gewinnt 100 Euro!");
```

Denken Sie darüber nach. Erweitern Sie das Fragment zu einem lauffähigen Programm mit Eingabe für `losNr`. Korrigieren Sie eventuelle Fehler.

2. Aufgabe

Warum liefert das folgende Programm widersprüchliche Auskünfte über die boolesche Variable b?

Quellcode	Ausgabe

`class Prog {`
`  public static void main(String[] args) {`
`  boolean b = true;`
`  if (b = false)`
`  System.out.println("b ist false");`
`  else`
`  System.out.println("b ist true");`
`  System.out.println("\nKontr.ausg.: b ist "+b);`
`  }`
`}`
	`b ist true`
	
	Kontr.ausg.: b ist false

3. Aufgabe

Das folgende Programm soll Buchstaben nummerieren:

```
class Prog {
   public static void main(String[] args) {
      char bst = 'a';
      byte nr = 0;
      switch (bst) {
         case 'a': nr = 1;
         case 'b': nr = 2;
         case 'c': nr = 3;
      }
      System.out.println("Zu "+ bst + " gehoert die Nummer "+nr);
   }
}
```

Warum liefert es zum Buchstaben a die Nummer 3, obwohl für diesen Fall die Anweisung **nr = 1** vorhanden ist? Korrigieren Sie eventuelle Fehler.

4. Aufgabe

Wie oft wird die folgende **while**-Schleife ausgeführt?

```
public class While {
	public static void main(String[] args) {
		int i = 0;
		while (i < 100);
		{
			i++;
			if(i<100)System.out.print(i + ", ");
			else System.out.print(i);
		}
	}
}
```

Erst nachdenken. Eventuell ist es nach einem Test sinnvoll Eclipse neu zu starten.

5. Aufgabe

Erstellen sie ein Konsolprogramm welches folgendes leistet:

- Es wird eine Feldlänge eingelesen, um damit ein **float**-Feld dieser Länge anzulegen.
- Die Feldpositionen sollen mit Werten initialisiert werden, die bei 0 beginnen und für die nächste Position jeweils mit 0.5 inkrementiert werden - also 0, 0.5, 1.0, 1.5, ...
- Machen Sie eine Testausgabe jedes Feldelementes, um sicher zu stellen, dass die Initialisierung korrekt ist.
- Erstellen Sie diese Übung in den folgenden 3 Varianten der Wiederholung:
  a) **for**
  b) **while**
  c) **do**…**while**

Ein-/Ausgabeprinzip (Beispiel):

```
Feldlänge> 5                                        //Eingabe
for-Schleife: 0.0, 0.5, 1.0, 1.5, 2.0, 2.5          //Ausgabe
```

6. Aufgabe

Erstellen sie ein Konsolprogramm welches folgendes leistet:

- Es sollen zwei Felder (f1, f2) vom Typ **int** erstellt werden, die gleich lang sind
- Die Feldlänge wird eingegeben, um damit die Felder anzulegen.
- Als nächstes werden in diese Felder beliebige Zahlen eingelesen
- Danach sollen die Feldpositionen mit gleichem Index verglichen und per Ausgabe festgehalten werden, ob f1[i] == f2[i], f1[i] < f2[i] oder f1[i] > f2[i] ist. Es soll der Index und das entsprechende Kürzel (==, <, >) ausgegeben werden.
- Es ist die Summe der Inhalte von f1 und f2 auszugeben

Hinweis: Verwenden Sie eine **for**-Schleife um die Zahlen einzulesen und eine zweite **for**-Schleife um die Index-Positionen zu vergleichen.

Ein-/Ausgabeprinzip (Beispiel):

```
Feldlänge> 3                        //Start Eingabe
f1[0]: 3
f2[0]: 4
f1[1]: 3
f2[1]: 3
```

```
f1[2]: 4
f2[2]: 3

3  <  4                          //Start Ausgabe
3 ==  3
4  >  3
```

7. Aufgabe

Erstellen Sie ein Konsolprogramm, das ein **char**-Feld der Länge 10 mit Zufallszeichen füllt. Die Zeichen sollen aus dem Bereich a-z (Codebereich 97 bis 122) bestimmt werden. Geben Sie diese Zeichen aus. Geben Sie auch die Codes und deren Summe aus.

Ausgabeprinzip:

Zufallszeichen: r, g, g, l, f, p, f, c, a, u

Codes mit Summe: 114 + 103 + 103 + 108 + 102 + 112 + 102 + 99 + 97 + 117 = 1057

8. Aufgabe

Erstellen sie ein Konsolprogramm welches folgendes leistet:

- Es sollen zwei beliebige Strings eingeben werden
- Untersuchen sie ob die Strings gleich lang sind und ggf. den gleichen Inhalt haben
- Machen Sie entsprechende Ausgaben
- Vereinigen Sie String1 mit String2 und geben Sie dann die einzelnen Zeichen des Gesamt-Strings aus (durch Leerzeichen getrennt)

Hinweis: Schauen Sie sich dazu die Operationen der Klasse String genau an.

Ein-/Ausgabeprinzip (Beispiel):

```
String1> hugo                    //Start Eingabe
String2> susi
                                 //Start Ausgabe
die Strings sind gleich lang mit unterschiedlichem Inhalt
h u g o s u s i
```

9. Aufgabe (Arbeiten mit Adressen)

Erstellen Sie ein Konsolprogramm, das schrittweise folgendes leistet:

1. Es wird eine Klasse Punkt erstellt; ein Punkt besteht aus einer x- und y-Koordinate.
2. Es soll ein Objekt der Klasse Punkt angelegt und in der Variablen *punkt* hinterlegt werden. Lesen für das Objekt x-/y-Koordinaten ein.
3. Erstellen Sie ein Feld *punkte* vom Elementtyp *Punkt* mit der Länge 3. Weisen Sie jedem Feldelement das erzeugt Objekt zu.
4. Erstellen Sie eine weitere Feldvariable des Typs *Punkt* – *punkteKopie* – und machen Sie die folgende Zuweisung: *punkteKopie = punkte;*
5. Geben Sie die Inhalte der Felder *punkte* und *punkteKopie* aus.
6. Lesen Sie neue x-/y-Koordinaten für das Objekt der Variablen *punkt* ein; geben Sie dann erneut die Inhalte der Felder *punkte* und *punkteKopie* aus.

- Hinweis zur Programmerstellung: Erstellen Sie eine öffentliche Klassen mit main-Methode zur Programmausführung. Erstellen Sie im gleichen File die Klasse Punkt als eigenen Typ.

Ein-/Ausgabebeispiel:

```
Punkt.x> 17
Punkt.y> 23
Punkte: Punkt[0](17, 23) Punkt[1](17, 23) Punkt[2](17, 23)
Kopie:  Punkt[0](17, 23) Punkt[1](17, 23) Punkt[2](17, 23)

Punkt.x> 31
Punkt.y> 11
Punkte: Punkt[0](31, 11) Punkt[1](31, 11) Punkt[2](31, 11)
Kopie:  Punkt[0](31, 11) Punkt[1](31, 11) Punkt[2](31, 11)
```

10. Aufgabe

Berechnen Sie für ein **int**-Feld die Summe und den Durchschnitt seiner Elemente. Dabei soll das Feld mit Zufallszahlen gefüllt werden. Folgender Ablauf ist vorgesehen:

- Als erstes wird die Feldlänge/Anzahl der Zufallszahlen eingelesen.
- Zusätzlich wird ein Intervall bestimmt (Einlesen von min und max), in dem die Zufallszahlen liegen sollen.
- Dann wird das Feld angelegt, mit den Zufallszahlen gefüllt, Summe und Durchschnitt der Feldinhalte berechnet, sowie Feld und Ergebnisse ausgegeben.

Ein-/Ausgabebeispiel:

```
Feldlänge> 8                                    //Eingabe
Intervall.min> 22                               //Eingabe
Intervall.max> 66                               //Eingabe

Feldelemente: 58 24 35 64 34 28 65 53           //ab hier Ausgabe
Summe: 361
Durchschnitt: 45.125 -> 361/8
```

11. Aufgabe – optional

Es soll ein Programm geschrieben werden, das einen Kassenbon auf der Konsole anzeigt, für drei verschiedene Artikel, die gekauft werden sollen. Dazu sind folgende Eingaben vorzunehmen bzw. Funktionalität bereitzustellen:

- Eingabe: Inhalt der Brieftasche
- Eingabe: Bezeichnung, Menge und Einzelpreis der jeweiligen Artikel
- Es wird der Gesamtpreis errechnet. Übersteigt der Gesamtpreis den Inhalt der Brieftasche, wird auf den fehlenden Betrag hingewiesen. Ansonsten wird ein Kassenbon ausgegeben, der die gekauften Artikel mit Anzahl und Preis sowie am Ende den Gesamtpreis auflistet.

Ein-/Ausgabebeispiele	
Beispiel 1	**Beispiel 2**
Inhalt Brieftasche> 50 *Bezeichnung1> Wurst* *Menge1> 5* *Preis1> 3.75* *Bezeichnung2> Brot* *Menge2> 7* *Preis2> 2.30* *Bezeichnung3> Wein* *Menge3> 2* *Preis3> 17.95* Für diesen Einkauf fehlen 20,75 €	*Inhalt Brieftasche> 75* *Bezeichnung1> Wurst* *Menge1> 5* *Preis1> 3.75* *Bezeichnung2> Brot* *Menge2> 7* *Preis2> 2.30* *Bezeichnung3> Wein* *Menge3> 2* *Preis3> 17.95* ---------- Wurst: 5 * 3,75 --> 18,75 Brot: 7 * 2,30 --> 16,10 Wein: 2 * 17,92 --> 35,84 ---------- Gesamt: 70,69 € Gegeben: 75,00 € Zurück: 4,31 €