Aufgabe 1: Datentypen und Typecasting

Geben Sie das Ergebnis für folgende mathematischen Ausdrücke im entsprechenden Datentyp in Java an. Beachten Sie hierzu die Folie 14 der Präsentation.

Hinweis:

Bei der Auswertung der Ausdrücke gilt die allgemeine "Punkt-vor-Strich" Regel. Die Auswertungsreihenfolgekann, wie in der Mathematik auch, durch Klammern geändert werden.

Beispiel: `2.5 * 2 + 3 = 8.0 (Das Ergebnis ist eine Kommazahl)`

```
2.5 * (2 + 3) =                 12.5 (Kommazahl)
(int) 2.5 * 2 + 3 =             7    (Ganzzahl)
(int) 2.5 * (2 + 3) =           10   (Ganzzahl)
(int) (2.5 * 2 + 3) =           8    (Ganzzahl)
(int) (2.5 * (2 + 3)) =         12   (Ganzzahl)
(int) 2.5 * 2 + (float) 3 =     7.0  (Kommazahl)
```

Aufgabe 2: Namenskonventionen für Methoden

Auf Folie 19 sehen Sie eine "Konvention" für die Namen von Zugriffsmethoden.

- Was bedeutet in diesem Zusammenhang "Konvention"?
  Eine Konvention ist eine Richtlinie
- Warum ist eine solche Konvention sinnvoll?
  Dient zur besseren Verständlichkeit, Lesbarkeit.
- Geben Sie Signaturen für Zugriffsmethoden auf die folgenden Attribute gemäss der Konvention an.
  - `String vorname;` `String getPrename()`
  - `float stunden;` `float getHour()`
  - `int personalNummer;` `int getPersonNumber()`
  - `Object meinObjekt;` `Object getOwnObject()`
- Kann man für Mutator Methoden auch Namenskonventionen definieren? Begründen Sie.
  Ja, die gleichen wie für Methoden.

Aufgabe 3: Fehlermeldungen

Bei den Aufgaben 2.25 und 2.54 mussten Sie Fehlermeldungen notieren.

- Vergleichen Sie Ihre Notizen.

```
missing return statement
unreachable statement
```

- Leiten Sie aus den Fehlermeldungen je eine Regel/ einen Merksatz ab.

missing return statement

Wenn eine Methode mit einem Rückgabewert definiert wird, muss immer ein Return Befehl darin vorhanden sein.

unreachable statement

Nach einem Return-Befehl dürfen keine weitere Befehle in dieser Methode folgen.

Aufgabe 4: Lernaufgabe zu Switch

Bearbeiten Sie zu zweit die Lernaufgabe zu *switch*. In dieser Lernaufgabe erarbeiten Sie sich selbstständig die Funktionsweise der *switch* Anweisung.
Benutzen Sie dazu das Lehrbuch Anhang D.1 und die Datei *Auswahl.jar* vom ILIAS.

Aufgabe 4.1 (###WICHTIG ###)

Betrachten Sie sich folgende Methode:

```
public void ausgabe (int wert)
{
  System.out.println ();
  System.out.println ("aktueller Parameter: " + wert);
  switch (wert)
  {
          case 1: System.out.println ("eins");
                  break;
          case 2: System.out.println ("zwei");
                  break;
          case 3: System.out.println ("drei");
                  break;
          default: System.out.println ("anderer Wert");
                  break;
  }
}
```

Lesen Sie die Seiten 475 bis 477 (Abschnitt switch) in Ihrem Buch.
Hinweis (zum Buch): *expression* beim *switch* bezeichnet einen Wert vom Datentyp *byte, char, short* oder *int*.
Stellen Sie eine Vermutung an über die Ausgabe der Methode, wenn Sie beim Methodenaufruf folgende Werte übergeben:

-1	*default: System.out.println ("anderer Wert");*	*anderer Wert*
0	*default: System.out.println ("anderer Wert");*	*anderer Wert*
1	*case 1: System.out.println ("eins");*	*eins, zwei, drei, anderer Wert*
3	*case 3: System.out.println ("drei");*	*drei, anderer Wert*

Aufgabe 4.2:

Holen Sie sich die Klasse Auswahl von ILIAS. Sie ist in Form eines .jar Files gespeichert.
- Speichern Sie diese Datei ab.
- Verwenden Sie in BlueJ das Kommando Project ➔ Open Non BlueJ
- Geben Sie hier das abgespeicherte .jar File an.

Sie erstellen so ein BlueJ Projekt, das Ihnen die Klasse Auswahl zur Verfügung stellt.
Überprüfen Sie Ihre Vermutungen aus Aufgabe 1, indem Sie ein Objekt der Klasse *Auswahl* erzeugen und die Methode *ausgabe* mit oben genannten Werten aufrufen. Was passiert?
(Wenn Sie überrascht sind, lesen Sie nochmals die Seiten 475 bis 477 durch und suchen Sie nach einer Erklärung.)

Aufgabe 4.3:

Verändern Sie den Code der Methode *Ausgabe* so, dass sie folgendes Verhalten aufweist:

Wert des aktuellen Parameters	Ausgabe
1	eins
2	zwei
3	drei
anderer Wert	anderer Wert

```
/**
 * In Abhängigkeit des übergebenen Wertes wird erfolgt die Ausgabe eines Textes.
 *
 * @param  wert Die Methode erhält einen ganzzahligen Wert.
 */
public void ausgabe(int wert)
{
    System.out.println();
    System.out.println("aktueller Parameter: " + wert);
    switch(wert)
    {
        case 1: System.out.println ("eins");
                break;
        case 2: System.out.println ("zwei");
                break;
        case 3: System.out.println ("drei");
                break;
        default: System.out.println ("anderer Wert");
                break;
    }
}
```

Aufgabe 4.4:

Beschreiben Sie in Ihren eigenen Worten die Funktionsweise von *break*.

Nach einem Break werden keine weiteren Cases von dieser Switch Anweisung ausgeführt.

Aufgabe 4.5:

Ergänzen Sie die Klasse *Auswahl* um eine Methode *ausgabeTag* mit folgender Signatur:

```
ausgabeTag(int zahl)
```

Wenn Zahl einen Wert von 1 bis 7 hat, soll entsprechend dem Wert "Montag", "Dienstag", ... "Sonntag" ausgegeben werden. Im anderen Fall ist "falscher Wert" auszugeben.

```
/**
 * Rückgabe des Wochentages
 *
 * @param  zahl Wochentag als Zahl (Montag = 1)
 */
public void ausgabeTag(int zahl)
{
  switch(zahl) {
     case 1:  System.out.println("Montag");
              break;
     case 2:  System.out.println("Dienstag");
              break;
     case 3:  System.out.println("Mittwoch");
              break;
     case 4:  System.out.println("Donnerstag");
              break;
     case 5:  System.out.println("Freitag");
              break;
     case 6:  System.out.println("Samstag");
              break;
     case 7:  System.out.println("Sonntag");
              break;
     default: System.out.println("falscher Wert");
              break;
  }
}
```

Ergänzen Sie die Klasse *Auswahl* um eine Methode *ausgabeTag2* mit folgender Signatur:

```
ausgabeTag2(int zahl)
```

Wenn Zahl einen Wert von 1 bis 5 hat, soll "Werktag" ausgegeben werden. Bei einem Wert von 6 oder 7 wird "Wochenende" ausgegeben. Im anderen Fall ist "falscher Wert" auszugeben.

```
/**
 * Rückgabe ob Werktag oder Wochenende
 *
 * @param  zahl Werktag 1-5, Wochenende 6,7
 */
public void ausgabeTag2(int zahl)
{
  switch(zahl) {
     case 1:
     case 2:
     case 3:
     case 4:
     case 5:  System.out.println("Werktag");
              break;
     case 6:
     case 7:  System.out.println("Wochenende");
              break;
     default: System.out.println("falscher Wert");
              break;
  }
}
```

Aufgabe 5: Weitere Aufgaben aus dem Buch [optional]

Bearbeiten Sie die Aufgaben ab Exercise 2.59 in Ihrem Buch.

2.59:

name: getCode

return type: String

2.60:

name: setCredits

name parameter: creditValue

type parameter: int

2.61:

```
/**
 * Klasse Person
 *
 * @author Felix Rohrer
 * @version 1.0
 */
public class Person
{
   [...]
}
```

2.62:

```
private String name;
private int age;
private String code;
private int credits;
```

2.63:

```
/**
 * Konstruktor Module
 */
public Module(String moduleCode)
{
   code = moduleCode;
}
```

2.64:

```
    /**
     * Konstruktor Person
     */
    public Person(String myName, int myAge)
    {
       name = myName;
       age  = myAge;
    }
```

2.65:

```
public int getAge()
{
  return age;
}
```

2.66:

```
public String getName()
{
  return name;
}
```

2.67:

```
public void setAge(int newAge)
{
  age = newAge;
}
```

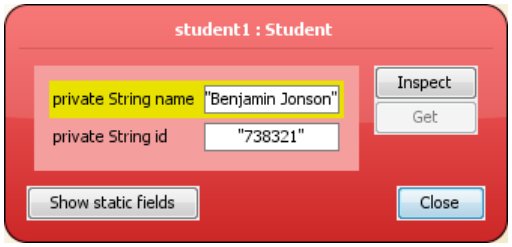
2.68:

```
public void printDetails()
{
  System.out.println("The name of this person is " + name);
}
```

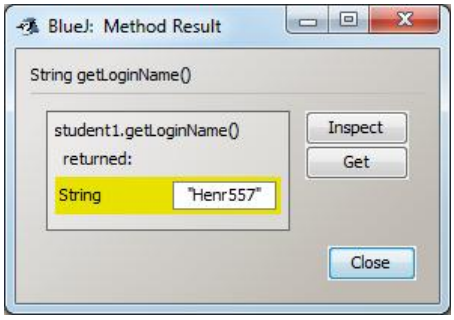
2.69:

```
/**
 * Write a description of class Student here.
 *
 * @author  Felix Rohrer
 * @version 2011-10-13v0
 */
public class Student
{
    // instance variables
    private String name;
    private String id;

    /**
     * Constructor for objects of class Student
     *
     * @param   fullName, studentID
     */
    public Student(String fullName, String studentID)
    {
        name = fullName;
        id   = studentID;
    }
}
```



2.70:



2.71:

```
java.lang.StringIndexOutOfBoundsException: String index out of range: 4
  at java.lang.String.substring(String.java:1934)
  at Student.getLoginName(Student.java:27)
```

Der String "name" ist zu kurz. Er müsste mindestens 4 Zeichen lang sein.

2.72:

```
    /**
     * Constructor for objects of class Student
     *
     * @param   fullName, studentID
     */
    public Student(String fullName, String studentID)
    {
        if (length(fullName) < 4) {
            System.out.println("Der Name ist zu kurz, bitte mindestens 4 Zeichen eingeben!");
        }

        if (length(studentID) < 3) {
            System.out.println("Die ID ist zu kurz, bitte mindestens 3 Zeichen eingeben!");
        }

        name = fullName;
        id   = studentID;
    }

    public int length(String myString)
    {
        return myString.length();
    }
```

2.73:

```
    public String getLoginName()
    {
        String loginName;

        if (length(name) < 4) {
           loginName = name;
        } else {
           loginName = name.substring(0,4);
        }

        if (length(id) < 3) {
           loginName += id;
        } else {
           loginName += id.substring(0,3);
        }

        return loginName;
    }
```

2.74:

99 + 3	*102*
“cat” + “fish”	*catfish*
“cat” + 9	*cat9*
9 + 3 + “cat”	*12cat*
“cat” + 3 + 9	*cat39*
“catfish”.substring(3,4)	*f*
“catfish”.substring(3,8)	*Error: String index out of range: 8*

substring`(int beginIndex, int endIndex)`
Returns a new string that is a substring of this string.

2.75:

2.76:

2.77:

2.78:

2.79:

2.80:

2.81:

2.83:

2.84: