Allgemeine Informationen: Beispiele, die über das Abgabesystem abzugeben sind, haben in eckigen Klammern die abzugebende(n) Datei(en) vermerkt. Diese Beispiele sind bis Donnerstag, den 10.10., 8:00h abzugeben. Die anderen Beispiele sind bis zur Übungseinheit am Donnerstag, den 10.10., vorzubereiten. **Achtung:** Legen Sie Klassen und Methoden nur dann als *public* an, wenn dies explizit gefordert ist. Ansonsten erkennt das Abgabesystem die abgegebenen Klassen/Methoden nicht.

Testen Sie alle Programme entweder im BlueJ-Objektbrowser oder durch mithilfe von Testklassen!

Beispiel 1.1
Schreiben Sie eine Klasse *Sparkassa* mit parameterlosem Konstruktor, der eine leere Sparkassa erstellt.

Schreiben Sie weiters eine Methode

```
public void einzahlen(int betrag),
```

die den übergebenen Betrag in die Kassa einzahlt.
Mit

```
public int leeren()
```

soll die Kassa geleert und der angesparte Betrag zurückgegeben werden.

Beispiel 1.2
Schreiben Sie eine Klasse *Revolver* mit Konstruktor

```
Revolver(int anzSchuesse),
```

der einen geladenen Revolver anlegt, der *anzSchuesse* abgeben kann.

Weiters soll es folgende Methoden geben:

```
boolean gibSchussAb()
```

soll –falls möglich– einen Schuss abgeben. In diesem Fall soll die Methode *true* zurückgeben, sonst *false*.

```
void laden()
```

soll den Revolver wieder aufladen.

Beispiel 1.3
In Programmen von Studierenden findet man öfters *if-else*-Anweisungen mit leerem Anweisungsteil, z.B.

```
if(n<k){

}else{i++;}
```

Wie lässt sich das einfacher formulieren?

Beispiel 1.4 ***[BoolescherAusdruck.java]***
Schreiben Sie eine Klasse *BoolescherAusdruck* mit parameterlosem Konstruktor und einer Methode

```
boolean sindGleich(boolean a, boolean b),
```

die genau dann *true* zurückgibt, wenn *a* und *b* beide *true* sind oder wenn *a* und *b* beide *false* sind.

Beispiel 1.5
Machen die beiden folgenden Programmfragmente dasselbe? Warum (nicht)?

```
if(n<k){
    return true;
}else{
    return false;
}
```

```
if(n<k){
   return true;
}
return false;
```

Machen die beiden folgenden Programmfragmente dasselbe? Warum (nicht)?

```
if(n<k){
    i++;
}else{
    i--;}
```

```
if(n<k){
   i++;
}
i--;
```

Beispiel 1.6 ***[Heizungssteuerung.java]***
Schreiben Sie (zum Aufwärmen) eine Klasse *Heizungssteuerung* mit Konstruktor

```
public Heizungssteuerung(int min, int max),
```

der eine Heizungssteuerung anlegt, die eine Heizung zwischen der minimalen Temperatur *min* und der maximalen Temperatur *max* steuert. Zu Beginn soll die Temperatur auf die Minimaltemperatur gestellt werden. Die Methoden

```
public void waermer(),
public void kuehler()
```

sollen die Temperatur um eine voreingestellte Schrittweite erhöhen bzw. verringern. Die Schrittweite soll zu Beginn auf 5 gesetzt werden. Wird durch das Erhöhen der Temperatur der Maximalwert überschritten, soll die Temperatur auf den Maximalwert gesetzt werden. Analoges gilt für das Verringern der Temperatur. Die Methode

```
public int getTemperatur()
```

soll die aktuell eingestellte Temperatur zurückgeben. Mit der Methode

```
public void setSchrittweite(int s)
```

kann die Schrittweite auf den neuen Wert *s* gesetzt werden, vorausgesetzt der übergebene Wert *s* ist größer als 0.

Beispiel 1.7 ***[Zeitpunkt.java]***
Schreiben Sie eine Klasse *Zeitpunkt*, die einen Zeitpunkt mit Datum und Uhrzeit festlegt, mit einem Konstruktor

```
Zeitpunkt(int jahr, int monat, int tag, int stunde, int minute),
```

der einen entsprechenden Zeitpunkt erstellt. Schreiben Sie weiters eine Methode

```
String getZeitpunkt(),
```

die den Zeitpunkt als String in der Formatierung „2018/11/21 11:11“ zurückgibt.
(Beachten Sie, dass führende Nullen auszugeben sind, also z.B. „2018/09/01 08:05“.)

Beispiel 1.8
Schreiben Sie eine Klasse *Termin* mit Konstruktoren

```
Termin(Zeitpunkt einZeitpunkt, String artDesTermins),
Termin(int jahr, int monat, int tag, int stunde, int minute,
                                          String artDesTermins),
```

die einen Termin für den jeweils angegebenen Zeitpunkt mit dem Zusatztext *artDesTermins* erstellen. (Verwenden Sie die Klasse Zeitpunkt aus Beispiel 1.7.) Schreiben Sie in die Klasse *Termin* weiters eine Methode

```
String getTermin(),
```

die den Termin in der Formatierung „2019/10/10 09:15 : Vorlesung IT I“ ausgibt.

Beispiel 1.9
Schreiben Sie eine Klasse *Fluessigkeit* mit Konstruktor/Methoden

```
Fluessigkeit(double menge, double temperatur),
double getMenge(),
double getTemperatur(),
void fuegeHinzu(Fluessigkeit andereFluessigkeit).
```

Die Methoden *getMenge* und *getTemperatur* sollen jeweils die vorhandene Menge der Flüssigkeit bzw. ihre Temperatur zurückgeben. Die Methode *fuegeHinzu* fügt eine andere Flüssigkeit hinzu, wodurch sich Menge und Temperatur der ursprünglichen Flüssigkeit nach folgender Regel ändern: Wenn *m1* und *t1* die ursprüngliche Menge und Temperatur der Flüssigkeit sind, *m2* und *t2* jene der anderen Flüssigkeit, dann ist die neue Menge *m1+m2* und die neue Temperatur (*t1*m1 + t2*m2*)/(*m1+m2*).

Beispiel 1.10
Schreiben Sie eine Klasse *Stromgenerator* mit Konstruktor

```
Stromgenerator(int maxLeistung),
```

der einen Generator mit entsprechender Maximalleistung erstellt. Weiters gebe es folgende Methoden:

```
double etLeistung()
```

gibt die momentane Leistung zurück. (Diese soll zu Beginn 0 sein.)

```
void setLeistung(double anteil)
```

setzt die Leistung auf *anteil* % der Maximalleistung.

Schreiben Sie weiters eine Klasse *Kraftwerk* mit Konstruktor

```
Kraftwerk(double maxLeistung, double startLeistung),
```

der ein (kleines) Kraftwerk mit zwei Generatoren mit angegebener Maximalleistung anlegt. Zu Beginn soll die Gesamtleistung des Kraftwerks auf *startleistung* gesetzt werden. Rufen Sie dazu im Konstruktor die Methode *setGesamtleistung* (s.u.) auf.

Schreiben Sie weiters folgende beiden Methoden in *Kraftwerk*:

```
double getGesamtleistung()
```

gibt die momentane Gesamtleistung des Kraftwerks zurück.

```
void setGesamtleistung(double leistung)
```

setzt die Gesamtleistung des Kraftwerks auf den angegebenen Wert. Falls *leistung* über dem erreichbaren Wert liegt, soll die Gesamtleistung auf den maximal möglichen Wert gesetzt werden. Falls *leistung* kleiner als 0 ist, soll die Gesamtleistung auf 0 gesetzt werden.