Eigenes Programm erstelle und übersetzen

- Erstelle ein neues Projektverzeichnis namens "welcome-java" (im Ordner programs)
- In diesem Verzeichnis soll eine Quelltextdatei mit dem Namen "WelcomeJava.java" erstellt werden
- In der Quelltextdatei WelcomeJava.java soll der Text "Welcome to Java" auf der Kommandozeile ausgegeben werden.
- Übersetze die Quelltextdatei mit dem Java Compiler und führe sie mit dem Java Launcher aus.

Welche Fehler begeht der Entwickler hier?

Ein Entwickler legt eine neue Java-Quelltextdatei mit dem Namen `MyApp.java` an und füllt sie mit folgendem Inhalt:

```
public class App {

}
```

Anschließend will der Entwickler seine Quelltextdatei mit dem Java Compiler `javac` übersetzen. Dazu gibt er folgendes Kommando ein:

```
javac MyApp.java
```

Der Compiler meldet einen Fehler. Woran kann das liegen? Welche Änderungen sind am Quelltext vorzunehmen, damit der Compiler ein Kompilat erzeugen kann?

Der Entwickler hat seinen Fehler im Quelltext behoben. Nun will er sein Programm mit Hilfe des Java Launchers `java` ausführen. Dazu tippt er folgendes Kommando ein:

```
java MyApp.class
```

Abermals erhält der Entwickler einen Fehler. Was hat er falsch gemacht? Wie müsste das Kommando stattdessen lauten?

Der Entwickler bemerkt seinen Fehler und ändert das Kommando entsprechend ab. Doch jetzt erhält er plötzlich einen weiteren Fehler. Angeblich findet der Java Launcher keine `main` Methode. Was hat es damit auf sich? Wie lässt sich das Problem beheben?

Welche Ergebnisse liefern die folgenden Zuweisungen?

Teste folgende Anweisungen in der JShell. Überlege dir vorher, welches Ergebnis zu erwarten ist und prüfe erst anschließend, ob deine Vermutung richtig war. Kannst du die Ergebnisse begründen?

Hinweis: Manche Anweisungen sind syntaktisch falsch. Welche sind es und warum schlagen sie fehl?

Bemerkung: Die JShell startest du mit dem Befehl `jshell`. Du kannst die JShell verlassen, indem du den Befehl `/exit` eintippst und mit Taste Enter bestätigst.

```
int a = 1703;
int b = 0703;
int c = 4 / 8;
int d = 0xCAFE;
int e = Integer.MIN_VALUE - 1;
int f = Integer.MAX_VALUE + 1;
int g = 07709;
int h = -2e3;
byte i = 120 + 9;
byte j = 0b1000_0000;
byte k = (byte)(0b1000_0000);
byte l = (byte)(0xFF);
byte m = 1;
byte n = 2;
byte o = m + n;
byte p = (byte)(m + n);
int q = 4 / 3;
int r = 5 / 2.5;
double s = 1 / 2;
double t = 1.0 / 2;
double u = (double)(1 / 2);
double v = 1 / 3.0;
float w = 1;
float x = 0.5;
float y = 0.5f;
float z = 3 * 5 / 2 + 2.5f;
float aa = 1.0 + 2.5f;
float ab = 1.0f + 2.5f;
double ac = 1.0 + 2.5f;
char ad = 'c';
char ae = "c";
char af = 0xff;
int ag = (int)('U');
char ah = 'ab';
char ai = '\n';
char aj = '"';
char ak = '\'';
boolean al = false;
boolean am = 1;
boolean an = 0;
boolean ao = (boolean)(1 + 1);
boolean ap = 4 > 3;
boolean aq = 4 == 4;
```

Wie lauten die Wrapper Klassen der primitiven Datentypen?

Finde heraus, wie die Wrapper-Klassen der primitiven Datentypen `byte`, `short`, `char`, `int`, `float`, `double` und `boolean` heißen. Nutze dazu die JDK API Documentation.