## Java-Programme ausführen

Java-Programme bestehen meist aus mehreren Dateien: der Quellcode, der über mehrere Klassen verteilt ist, der kompilierte Quellcode (.class-Dateien) und ggf. weitere Dateien, die z.B. Informationen über graphische Oberflächen enthalten.

Um ein Java-Programm auszuführen, gibt es mehrere Möglichkeiten:

- Die Haupt-Methode mit **BlueJ** (oder einer anderen Entwicklungsumgebung) ausführen
- Die Haupt-Methode von der **Konsole** aus mit dem Befehl „java <class-Datei>“ aufrufen
- Das Projekt zu einer **JAR-Datei** zusammenfassen, und diese dann ausführen

Die dritte Möglichkeit (JAR-Datei) ist für den Benutzer die komfortabelste:
Man benötigt weder BlueJ, noch muss man aus den vielen Dateien eines BlueJ-Projekts die richtige herausfinden und einen umständichen Befehl auf der Konsole ausführen.
Außerdem kann man eine JAR-Datei einfach (z.B. als Download) an andere weitergeben.

## JAR = Java Archive

Eine JAR-Datei enthält eine oder mehrere kompilierte Java-Klassen.

Man kann eine JAR-Datei entweder als **Bibliothek** verwenden:
Dann können die Java-Klassen, die in der JAR-Datei enthalten sind, von anderen Programmen benutzt werden. Die GameWindow-Bibliothek ist zum Beispiel in einer JAR-Datei enthalten. Auch sämtliche Klassen, die von der Java-Klassenbibliothek zur Verfügung gestellt werden (z.B. String, Color, JFrame), sind in JAR-Dateien zusammengefasst.

Man kann eine JAR-Datei aber auch als **ausführbares Programm** verwenden.
Im Informatikunterricht benutzen wir z.B. UMLet, oder später das Programm JFLAP, die in Form einer JAR-Datei zur Verfügung stehen.

## Ausführbare JAR-Datei

Wenn eine JAR-Datei ein ausführbares Programm enthalten soll, MUSS eine der Klassen eine besondere Methode enthalten, und zwar die **main-Methode** – damit bei Ausführung klar ist, wo das Programm beginnt.

Die main-Methode muss dabei auch ein besonderes Format einhalten. Hier ein Beispiel:

```
public class Hauptklasse
{
    ...
    public static void main(String[] args)
    {
        ...
    }
}
```

Das Schlüsselwort „**static**“ bedeutet:

Normalerweise können Methoden nur über ein Objekt ausgeführt werden:

```
objekt.methode()
```

Eine „statische Methode“ kann direkt ausgeführt werden, ohne dass ein Objekt der Klasse erstellt wird. Und das ist in diesem Fall wichtig, denn Objekte können erst erstellt werden, nachdem das Programm schon gestartet wurde – aber um es zu starten muss eine Methode ausgeführt werden. (Das ist ein bißchen wie das Henne-Ei-Problem: Ein Huhn kommt aus einem Ei, aber um ein Ei zu erhalten braucht man ja ein Huhn.)

Der Parameter **String[] args** bedeutet:

Falls man das Programm von der Konsole startet, kann man dem Programm ggf. noch einen oder mehrere Parameter übergeben. Früher, als Computer noch hauptsächlich über eine Konsole bedient wurden, machten Programme häufig Gebrauch von dieser Möglichkeit.
Man führt auf der Konsole einen Befehl aus und setzt bestimmte Parameter als Optionen.

> Als Beispiel:
> Der Befehl `dir /a /p` zeigt den Inhalt des aktuellen Ordners an.
> /a und /p sind zwei Parameter. /a bedeutet: alle Dateien (auch versteckte) anzeigen.
> /p bedeutet: nicht alles auf einmal, sondern Seite für Seite anzeigen.

Um unsere Java-Programme auszuführen benötigen wir in der Regel jedoch keine Parameter für void main(), insofern braucht uns String args[] nicht weiter zu interessieren.

### Aufgabe von static void main(String[] args)

Da bisher noch kein Objekt erzeugt worden ist, ist die Aufgabe der statischen main-Methode, das erste Objekt des Programms zu erzeugen, und für dieses erste Objekt eine Methode auszuführen, die dann das ganze Programm in Gang setzt.

**Autor:** Christian Pothmann – cpothmann.de


**Quellen:** JAR-Archive: David Vignoni / iconking.com, Freigegeben unter GNU LGPL 3.0

## Vollständiges Beispiel

Am Beispiel eines Spiels (mit der Klasse GameWindow) ist hier dargestellt, wie das Programm durch die statische main-Methode in Gang gesetzt wird. Der Ablauf ist dann folgender:

1. static void main() wird ausgeführt (Parameter sind hier nicht erforderlich)
2. static void main() erzeugt ein Objekt der Hauptklasse Spiel
3. Mit dem Objekt der Klasse Spiel kann die Methode run() ausgeführt werden.
   Diese enthält den Game-Loop und damit das eigentliche Programm.

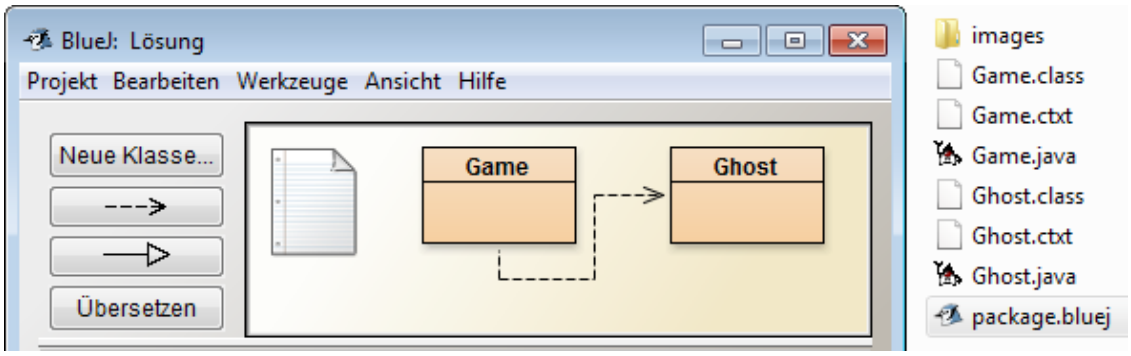
```
public class Spiel
{
   private GameWindow window;
   private GameImage  image;

   // 2. Konstruktor: erzeugt Unter-Objekte
   public Spiel()
   {
      window = new GameWindow(800, 600, "Spiel");
      image  = new GameImage("./images/bild.png");
   }

   // 3. Haupt-Ablauf des Programms
   public void run()
   {
      while (true)
      {
         window.clear();
         window.drawImage(image, 10, 10);
         window.paintFrame();
      }
   }

   // 1. Startpunkt: Erzeugt ein Objekt der Klasse Spiel
   // und führt mit diesem Objekt die run-Methode aus.
   public static void main(String[] args)
   {
      Spiel sp = new Spiel();
      sp.run();
   }
}
```

### Verständnisfrage

Warum programmiert man den Haupt-Ablauf nicht einfach in der statischen main-Methode?
D.h. warum braucht es den Umweg, in der main-Methode erst ein Objekt zu erzeugen, und dann darüber eine weitere Methode aufzurufen?

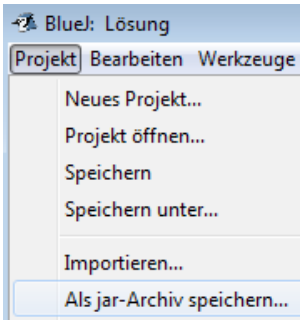
## JAR-Datei mit BlueJ erzeugen

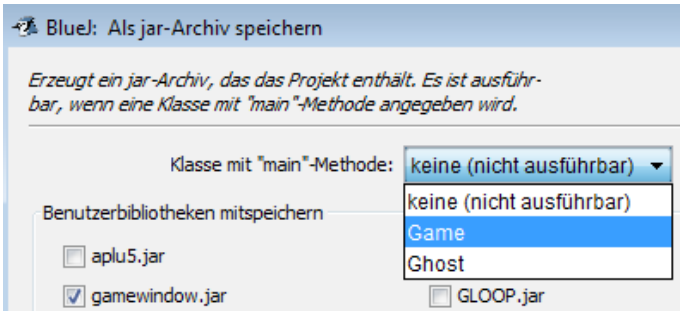
Als Beispiel wird ein Programm mit der GameWindow-Bibliothek verwendet.
Es besteht aus einer Hauptklasse „Game“ und einer Klasse für Pacman-Geister „Ghost“.
Der Ordner enthält außerdem einen Unterordner „images“ mit den Bilddateien für die Geister.




*Schritt 1*
Das Projekt muss zunächst **kompiliert** werden. Dabei erzeugt der Compiler aus den Quellcode-Dateien die class-Dateien „Game.class“ und „Ghost.class“.

*Schritt 2*
Mit BlueJ kann jetzt die JAR-Datei erstellt werden:
Wähle den Menüpunkt Projekt → **Als jar-Archiv speichern**.



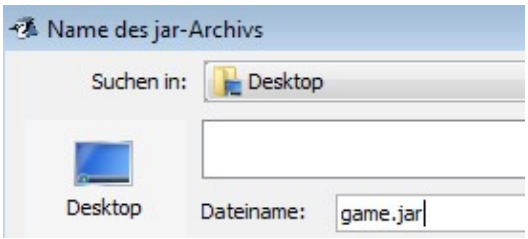



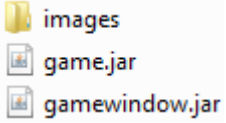

*Schritt 3*
Im folgenden Dialog muss man angeben, welche Klasse **static void main** enthält.

Außerdem mache ein Häkchen bei der **Bibliothek** gamewindow.jar, da diese vom Spiel benötigt wird.

*Schritt 4*
Gib einen **Dateinamen** für die JAR-Datei ein.
Er muss die Dateiendung .jar haben.

Es wird dann ein Ordner erstellt, der die Datei game.jar sowie die Bibliothek gamewindow.jar enthält.




*Schritt 5*
Das Spiel benötigt auch noch den **Unterordner mit den Bildern**.
Kopiere diesen in den Ordner mit den JAR-Dateien.
Durch einen **Doppelklick auf game.jar** kann das Spiel dann ausgeführt werden.