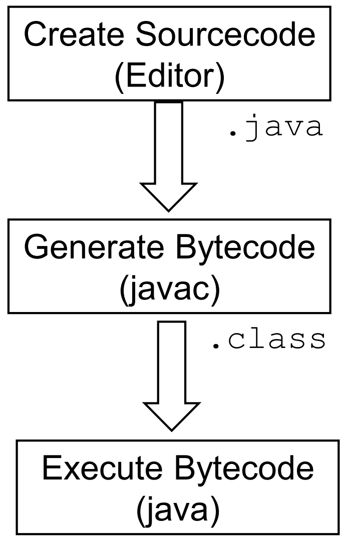
Java

Java Workflow



- In general, an arbitrary text editor will do. An IDE like Eclipse is more convenient but requires some time to get familiar with
- Example: We create the file `Hello.java`
- Generate Bytecode by means of `javac Hello.java`
- Result: The file `Hello.class` is generated. (If we defined such a class in our sourcecode).
- Execution by means of `java Hello`
- Java Bytecode is executed in a virtual machine

Hello World

A simple, executable class:

```
public class Hello {
  public static void main (String[] args){
    System.out.println("Hello World!");
  }
}
```

Hello World extended

Does this work?:

```
public class Hello {
  void f() {System.out.println("f");}

  public static void main (String[] args){
    System.out.println("Hello World!");
    f();
  }
}
```

Hello World extended

Solution 1 (if the method should belong to a class):

```
public class Hello {
  static void f() {System.out.println("f");}

  public static void main (String[] args){
    System.out.println("Hello World!");
    f();
  }
}
```

Hello World extended

Solution 2 (if the method should belong to objects):

```
public class Hello {
  void f() {System.out.println("f");}

  public static void main (String[] args){
    System.out.println("Hello World!");
    Hello h = new Hello();
    h.f();
  }
}
```

Two simple Classes

Our base-class:

```
class A{
  int x;
  void name(){System.out.println("Class A");}
}
```

Inheritance:

```
class B extends A{
  int y;
  B(){y = 99;}
  void name(){System.out.println("Class B");}
}
```

Using the Classes

Does this work?:

```
A a;
a.x = 7;
System.out.println(a.x);
```

Using the Classes

NO, we need to create an Object!:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
```

Using the Classes

We add a1, what output will we see?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
```

Using the Classes

Does this work?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = a;
```

Using the Classes

NO! Type A does not have everything of type B:

A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
~~b = a;~~

Using the Classes

Does this work?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
```

Using the Classes

This works as B is also of type A:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
```

Using the Classes

What will happen?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
System.out.println(a.x);
System.out.println(b.y);
System.out.println(a.y);
```

Using the Classes

Variable a is of type A which has no attribute y!:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
System.out.println(a.x); // displays 1
System.out.println(b.y); // displays 99
```
~~System.out.println(a.y);~~ // Error!

Using the Classes

What will happen?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
a1.name();
b.name();
a.name();
```

Using the Classes

What will happen?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
a1.name();
b.name();
a.name(); // displays class B (late binding)
```

Using the Classes

What will happen?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
B b1;
b1 = (B)a;
b1 = (B)a1;
```

Using the Classes

What will happen?:

```
A a = new A();
a.x = 7;
System.out.println(a.x);
A a1 = a;
a1.x = 9;
System.out.println(a.x);
B b;
b = new B();
b.x = 1;
a = b;
B b1;
b1 = (B)a; // works as referenced object of type B
b1 = (B)a1;// runtime error: cast not possible
```

Collections

Generic vs. typed:

```
public static void testAL(){
  ArrayList al = new ArrayList();
  al.add("x");
  al.add("y");
  String s = (String)al.get(0);
  System.out.println(s);

  ArrayList<String> al2 = new ArrayList<String>();
  al2.add("x");
  al2.add("y");
  s = al2.get(0);
  System.out.println(s);
}
```

Interfaces

Interface definition:

```
interface I{
  public void f1();
}
```

What will happen?:

```
public static void my_main (){
    I i = new I();
    i.f1();
}
```

Interfaces

Interface definition:

```
interface I{
  public void f1();
}
```

We can not create instances of interfaces!:

```
public static void my_main (){
    I i = ~~new I()~~; // error
    i.f1();
}
```

Interfaces

Interface definition:

```
interface I{
  public void f1();
}
```

Correct usage of the interface:

```
class C implements I{
  int n;
  public void f1(){
    System.out.println("f1");
  }
  public static void my_main (){
    I i = new C();
    i.f1();  }
}
```

Threads

- Threads allow concurrency
- A class using Threads can be derived from the class Thread or implement the interface Runnable
- Concurrent execution takes place in the method run()
- To execute run concurrent to the main control flow, it has to be invoked by means of start()!

Thread Example

A simple class using Threads

```
public class MyThread extends Thread{
 public void run(){
   try {
    System.out.println("OK1");
    sleep(2000);
    System.out.println("OK2");
  } catch ( Exception e) {System.out.println(
                 " Error1 :"+ e.getMessage());}
 }
 public static void main (String argv[]){
     MyThread t1 = new MyThread();
     t1.start();
     System.out.println("OKmain1");
 }
}
```