## Chapter 4: A Paradigm

- Program structure
- Connecting to the Java world
- Types
- Access modifiers
- Lifetime modifiers

## Example Program

```
import java.lang.*;

public class FirstProgram {
   public static void main (String [] args) {
      System.out.println("My first Java!");
   }
}
```

## How to run?

- javac FirstProgram.java
- ls FirstProgram.*

FirstProgram.class FirstProgram.java

- java FirstProgram

My first Java!

-

## Class name is IMPORTANT

- a handle for object creation (later)
- Filename = ClassName.java
  compiled file = ClassName.class

## Java program

= community of objects

Objects are instances of *classes*

A source program
=
Series of class descriptions

## Class definition

A class *header* plus a class *body*

```
public class FirstProgram {
   …
}
```

curly braces!
layout NOT significant => find your own style, but be consistent!

## Class body

a series of members of 2 types:

- Data fields: internal data of an object
- Methods: defines behavior

(FirstProgram: no data field, 1 method, *main* method = first method invoked by the interpreter)

## Method description

- Like class: header + body

*modifiers return-type name (arguments){*
*sequence-of-statements*
*}*
*public static void main (String[] args) { ..}*

- MUST be nested properly inside a class definition, lower case initial (convention!)

## Connecting to the JAVA world

*import java.lang.*;*

- makes this package visible,
- huge standard JAVA library (API = Applications Programming Interface)
- *System.out.println("My first Java!");*
  class *System*, data member *out*,
  behavior *println(),* one string argument

## Types

- Primitive data types: lowercase!
  *int newVar = 42;*
  int | float | double | boolean | void | …
  predefined, cannot define ourselves
- Classes: e.g. String
  *String name = "Fred Smith";*
  most types, can define ourselves

## boolean

Result of

- relational operator: <, <=
- logical operator: &&, ||, (or &, |)

strict left-to-right short-circuit evaluation:
SAFE: *(x > 0) && ((x/3) > 1.7)*

## Arrays

```
public class SecondProgram {
 public static void main (String [] args) {
   if (args.length > 0)
     System.out.println("Hello " + args[0]);
   else
     System.out.println("Hello everybody!");
   }
}
```

## Arrays cont.

- length not specified: *String[]*
- data member *length* holds actual size
- subscript operator: [i], e.g. *args[0],*
- zero-based: 0 .. n-1 if length=n

Strings: + means concatenation:

*System.out.println("Hello " + args[0]);*

## Three Access modifiers

control who can access/use classes, data members, and methods

- *public*: everybody,
- *private*: only inside the current class def
- *protected*: current class def and all subclasses (more in later chapters)

## Bank account example

```
public class BankAccount {
 private int balance = 0;
 public void deposit (int amount) {
   balance = balance + amount);
 }
 public void withdrawal (int amount) {
   balance = balance – amount;
 }
}
```

## Good practise

- data members are private
- (some) methods are public: provide services/interface to customers of the class

*public class FirstProgram { …*

*public static void main (String[] args) {..*

## Lifetime modifiers

*public static void main (String[] args) {..*

- Static parts are shared by all instances of a class, they come into existence when the class is loaded, even before any instance is created.
- Access by prefixing with class name:

  *System.out*

  *Math.log*

## Summary

- Import connects a program to libraries
- Program = sequence of class defs
- Class = header + body
- Body = data members + methods
- Method = header + body
- Method header = modifiers, return-type, name, arguments
- Method body = sequence of statements
- Access modifiers = public/private/protected
- static members: shared by all instances