


Lecture 1-2:

A Taste of Java and Object-Oriented Programming

Lecture outline

-  ■ Java background
- Java program example
- Basic syntax rules
- Program development life cycle
- A taste of object oriented programming
- Homework exercise 1

Java background

Brief history:

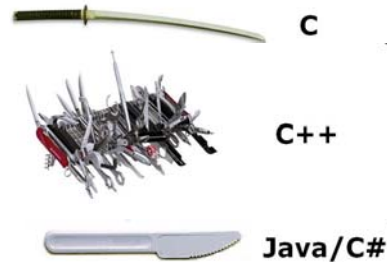
- 1995: invented by James Gosling at Sun Microsystems (1995)
 - Based on other languages, mainly C and C++
 - Original design objective: a programming language for the Internet: safety and portability
- 1996: IDC adopts Java as CS101 programming language
- 1999: Microsoft releases C#



Why did we adopt Java?

Because Java ...

- Is object oriented (OO)
- Encourages good programming habits
- Similar to C++, but simpler and more elegant
- Commercial
- Cool.



A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 3

Java program example

Task: Print the numbers 0 to 5

Algorithm:

```
i = 0;
while (i < 6)
  print i
  i = i + 1
```

Java implementation:

```
// prints the numbers 0 to 5
public class PrintSomeNumbers {
  public static void main(String[] args){
    // declare an integer variable and set it to 0
    int i = 0;
    while (i < 6) {
      // print the current value of i
      System.out.println(i);
      i = i + 1;
    }
    System.out.println("Done");
  }
}
```

A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 4

Java syntax elements (first approximation)

```
// prints the numbers 0 to 5
public class PrintSomeNumbers {
    public static void main(String[] args){
        // declare an integer variable and set it to 0
        int i = 0;
        while (i < 6) {
            // print the current value of i
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

keyword: public, class, static, void, while, int, String, System.out, println, Done

identifier: PrintSomeNumbers, args, i

comment: // prints the numbers 0 to 5, // declare an integer variable and set it to 0, // print the current value of i

symbol: {, }, (, <, >, [,], [,], {, }, ;, +, -, *, /, ...

numeric constant: 0, 6, 1

String constant: "Done"

"Words":

- Keywords
- Identifiers

Constants:

- Numbers
- Strings

Symbols:

- () [] { } , . ; + - * / ...

Comments

- Text beginning with //

A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 5

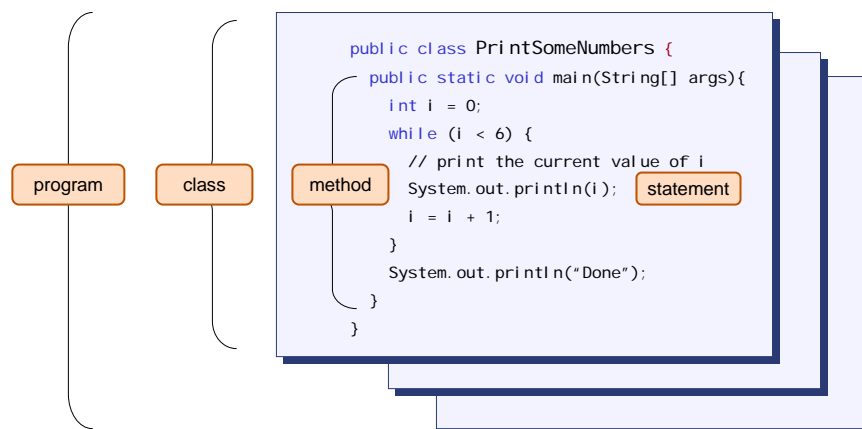
Java program structure

Program: a collection of one or more classes

Class: a collection of one or more methods, one of which must be named `Main()`

Method: a sequence of statements

Statement: `i = 0; System.out.println(i);` etc.



A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 6

White space

```
// prints the numbers 0 to 5
public class PrintSomeNumbers {
    public static void main(String[] args){
        // declare an integer variable and set it to 0
        int i = 0;
        while (i < 6) {
            // print the current value of i
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

Two equivalent programs

```
public class PrintSomeNumbers {public static void main(String[] args){int i=0; while (i <5){System.out.println(i); i=i+1; }System.out.println("Done"); }}
```

White space = comments and indentation (ignored by the compiler).

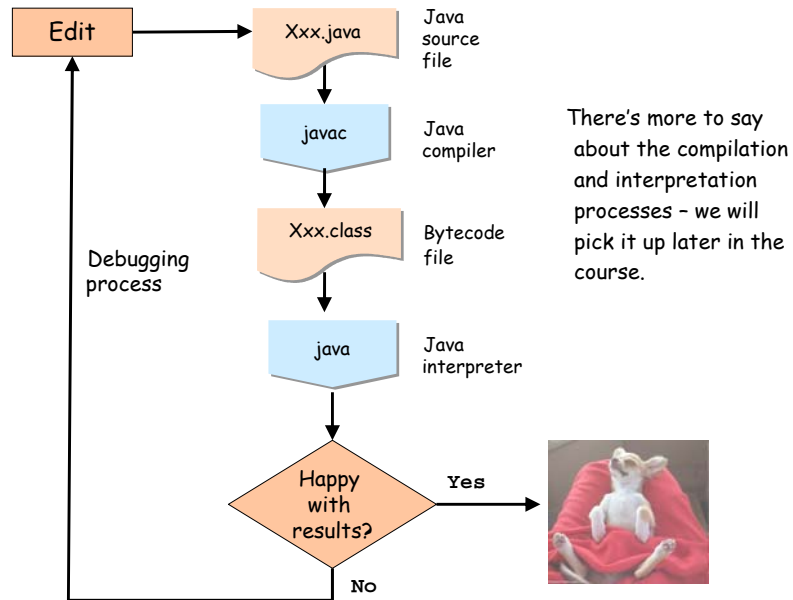
White space, left to the programmer's discretion, is used for readability

Important: Program readability is as important as program correctness!

Lecture outline

- Java background
- Java program example
- Basic syntax rules
- ➔ ■ Program development life cycle
- A taste of object oriented programming
- Homework exercise 1

Java program development life cycle



A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 9

The tools of the trade: basics

Simple text editor

```
PrintSomeNumbers.java - Notepad
File Edit Format View Help
public class PrintSomeNumbers{
    // prints the numbers 0 to 5
    public static void main(String[] args){
        // declare an integer variable and set it to 0.
        int i = 0;
        while (i < 6) {
            // print the current value of i
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

Compilation and execution:

```
C:\WINDOWS\system32\cmd.exe
D:\demo>javac PrintSomeNumbers.java
D:\demo>java PrintSomeNumbers
0
1
2
3
4
5
Done
D:\demo>_
```

A Taste of Java and OO programming, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 10

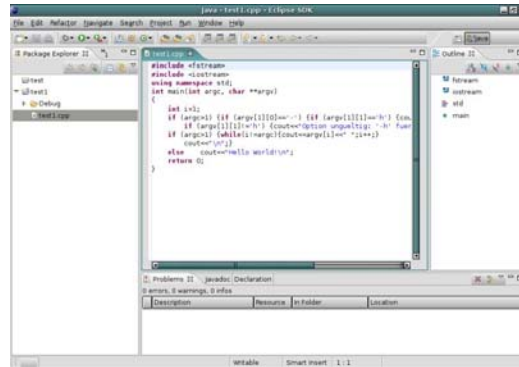
The tools of the trade: Integrated Development Environments

■ IDE: a software package that includes a combination of:

- Editor (programming-oriented)
- Compiler
- Debugger
- Project Manager

■ Some Commercial IDEs:

- Eclipse (open source)
- Visual Age (IBM)
- IntelliJ
- ...



Debugging

That's what you'll do most of the semester

Error types:

- **Compile-time errors:** mostly syntax violations; detected by the compiler
- **Run-time errors:** the program passes compilation, runs, but crashes
- **Logical errors:**
 - The program runs, doing something you didn't want it to do
 - The program runs, but should be improved for one reason or another.



Murphy's Law:
Anything that can possibly go wrong, does.

Errors are the portals of discovery
(James Joyce)

Lecture outline

- Java background
- Java program example
- Basic syntax rules
- Program development life cycle
- ➔ ■ A taste of object oriented programming
- Homework exercise 1

Abstraction and implementation

Turtle abstraction

A turtle is a turtle-like graphical image that moves on the screen under program's control.

When the turtle's tail is down, the movements are traced (drawn on the screen). When the tail is up, the movements are not traced.

The turtle is always facing a certain direction, and its tail is always either up or down.

We wish to be able to create new turtles (turtle objects) and give them movement and drawing instructions.



Turtle implementation

A public Java class, named `Turtle`, that implements the desired data and operations of a graphical turtle, as stated by the abstraction.

Using the Turtle

If a programmer wants to do turtle graphics, she can write a program that invokes the services of the `Turtle` class.

To do so, she will need access to the `Turtle` class **interface**.

Class interface

Class Turtle

Constructor Summary

Turtle ()
Constructs a new turtle.

Method Summary

void	disableDelay ()	Disables the delay of the turtle.
void	hide ()	Hides the turtle.
void	moveBackward (double units)	Moves the turtle backwards by a given number of units.
void	moveForward (double units)	Advances the turtle forwards by a given number of units.
void	show ()	Shows the turtle.
void	tailDown ()	Lowers the tail of the turtle.
void	tailUp ()	Raises the tail of the turtle.
void	turnLeft (int degrees)	Turns the turtle counter-clockwise.
void	turnRight (int degrees)	Turns the turtle clockwise.

- The Turtle class is a black box: we have no access to its code
- However, we do have access to the Turtle interface, also known as API
- Among other things, the API specifies which operations can be invoked on turtle objects
- If you want to write a program that does turtle graphics, all you need is the Turtle API.

Using the Turtle

Class Turtle

Constructor Summary

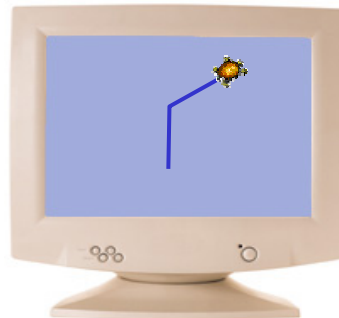
Turtle ()
Constructs a new turtle.

Method Summary

void	disableDelay ()	Disables the delay of the turtle.
void	hide ()	Hides the turtle.
void	moveBackward (double units)	Moves the turtle backwards by a given number of units.
void	moveForward (double units)	Advances the turtle forwards by a given number of units.
void	show ()	Shows the turtle.
void	tailDown ()	Lowers the tail of the turtle.
void	tailUp ()	Raises the tail of the turtle.
void	turnLeft (int degrees)	Turns the turtle counter-clockwise.
void	turnRight (int degrees)	Turns the turtle clockwise.

Turtle graphics example

```
public class TurtleDrawingDemo {  
    public static void main(String[] args){  
        Turtle leonardo = new Turtle();  
        leonardo.tailDown();  
        leonardo.moveForward(100);  
        leonardo.turnRight(60);  
        leonardo.moveForward(100);  
    }  
}
```



Object oriented programming

- In OOP, much of the programming activity evolves around creating and manipulating objects. For example, Leonardo is an object of type Turtle
- The objects can be created and managed by any class that anyone writes
- Therefore, an OO program can be described as a collection of classes which interact with each other
- Some of these classes are written by you; other classes are written by other programmers who you may or may not know
- For example, if someone wrote a class named ShoppingCart and made it public, any programmer who needs to develop shopping cart services can now use it

Some OOP advantages

- Code reuse
- Code consistency
- Divide and conquer
- Modularity.

Homework Exercise 1

- Play with a simple Java program
- Experience debugging
- Do some turtle graphics
- Further instructions: see the course web site.