Boaz Kantor
Introduction to Computer Science,
Fall semester 2010-2011
IDC Herzliya

Welcome, geeks!

Introduction to Java Programming

Plan for today:

1. Before we begin..
2. What is Java?
3. How to program?
4. Object Oriented Programming (OOP)
5. The basics of Java
6. Playing with turtles
7. Exercise #1

Before we begin..

- Course website and forum
- Homework submission
- The difference between a lecture and recitation
- Reminder: how to succeed in this course
- Vision:
  - Help students in becoming the best software engineers in the world.
  - Hard work, etiquettes, creativity, initiatives, fun!

Why computers?
Why programming languages?

Let's build a house together

So what's programming?

- Think of the people needed to build a house:
  - Planning: Architects, designers, engineers
  - Operations: Project planners, managers
  - Build: infrastructure, windows, heavy machinery, delicate work, etc.
- To build software you need:
  - The same people!! (with mild differences ☺)
- Java is the tool box of the builders

What can we do with Java?

- With Java we can build software
- Things to consider when building software:
  - Performance, efficiency
  - Security
  - Readability
  - Beauty
  - Scalability
  - Existing solutions, reusability

Sounds complicated?

It is.

The Concept of Object Oriented Programming

Object Oriented Programming

- Very similar to real life objects
- A simple concept:
  - Everything is an object
  - An object can do stuff. What can it do?
  - We don't care how it does it.
    - Unless we are the ones to program it..
- We will discuss OOP later in the course

The basics of Java

```
public class Program/class name {
      public static void main(String[] args) {

            Your algorithm goes here

      }
}
```

Until further notice, we only discuss the algorithm part.

Java basics, lesson #1

- The elements of a language
- Introduction to variables
- Introduction to loops
- Introduction to using objects

The elements of a language (1/2)

- Literals (values)
  - Numeric: `40, -12, 0, 4.17`
  - Textual: `'H', 'e', 'l', "Hello, world!"`
  - Boolean: `true, false`
- Expressions, made of operands (values) and operators
  - Arithmetic: `(12 + 6) /3`
  - Textual: `"Hello, " + "world!"`
  - Boolean:
    - `4 < 10, (41 / 4) == 10, 15 <= 20, 5 != -1, 'C' > 'A'`

The elements of a language (2/2)

- Statements:
  - Variable declaration: `float someVariable;`
  - Variable assignment: `someVariable = 10 / 2;`
  - Method call: `turtle.moveForward(100);`
  - Flow control: `if, while, foreach, …`
- Variables
- Classes & programs

Introduction to Variables

- A variable is a place in the computer memory (RAM) where we can store data.
- Each variable has a name that we set, so that we can refer to that place in memory whenever we want.
- Whenever a value is needed (e.g., in expressions) we can use a variable name instead.
- In Java, we need to pay attention to the variable type.

Syntax:

Declare a new variable:	`<data_type> <variable_name>;`
Assign a new value:	`<variable_name> = <value>;`
Can be done together:	`<data_type> <variable_name> = <initial_value>;`

Possible data types:

Whole numbers (integers):	`int, byte, short, long`
Floating points:	`float, double`
Boolean:	`boolean`
Objects:	`String, Turtle, MyFirstProgram`

Possible values:

Numeric literals:	`-12, 0, 54, 200000.68, 0.5`
String literals::	`"Hello, world!", "Ken sent me"`
Literal expressions:	`6 + 12.4, (55 / 2) * 17`
Variable expressions:	`var1, var1 * 2, var3 – var1, var1 + var2 + ..`
Boolean expressions:	`5 > -3, var1 <= var2, (var1 / var2) > var3`

Examples, primitive variables and expressions:

```
int var1 = 3;
int var2 = var1;
int var3 = var1 / var2;
int var4 = var1 – var2;
int var5 = var3 / var4;

boolean v6 = false;

var2 = 6;
var1 = var2 * var1;
var3++;
var4 = var4;
var2 = var1 / 2;

v6 = var2 > var1 + 5;
```

Introduction to loops

- Run the same statement over and over again.
- Stop when a condition is not met
- A very powerful tool!
  - But, with great power comes..
- The challenge: what condition to use

Syntax:

```
while (<boolean_expression>) {
        <loop_statements>
}
```

Possible loop statements:

Zero or more Java statements.
If zero statements, end with a semicolon:

```
        while (<boolean_condition>);
```

If 1 statement, no need for braces:

```
        while (<boolean_condition>)
                <loop_statement>;
```

Examples, simple 'while' loops:

```
int x = 0;
while (x < 10) {
        System.out.println("x = " + x);
        x = x + 1;
}
```

```
int var1 = 3;
int var2 = 5;
int var3 = var2;
while (var1 > 0) {
        var3 = var3 * var2;
        var1--;
}
```

1. Beware of endless loops!! Verify that the condition eventually evaluates to false.
2. Think of algorithms as standalone units.
   - Initialize var3 with 1 to make a 'power' algorithm.

Introduction to using objects

- We have many ready-made objects available for us.
- We will write our own objects in the future.
- To use an object, we must:
  - Learn its API (user manual)
  - Declare and initialize a variable ("instantiate")
  - Run the object's operations

Syntax:

Instantiation:

```
<class_name> <object_name> = new <class_name>();
<class_name> <object_name> = new <class_name>(<parameters>);
```

Running operations:

```
<object_name>.<operation>();
<object_name>.<operation>(<parameter>);
<object_name>.<operation>(<parameter1>, <parameter2>, ..);
```

Examples, simple object operations:

```
Turtle t = new Turtle();
t.moveForward(50);
t.turnRight(90);
t.tailDown();
t.moveBackward(100);
t.junmpTo(100, 200);
t.turnLeft(45);
t.moveForward(400);
```

Putting it all together

```
Turtle turtle = new Turtle();
t.tailDown();
int side = 0;
while (side < 4) {
        t.moveForward(200);
        t.turnRight(90);
}
t.hide();
```

```
int numberOfSides = 8;
int currentSide = 0;
int angle = 360 / numberOfSides;
Turtle t = new Turtle();
while (currentSide < numberOfSides) {
        t.moveForward(100);
        t.rurnLeft(angle);
}
```

Exercise #1

Due Wednesday, October 20, 16:00