

Lecture 2-2:
Using Classes and Objects

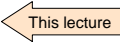
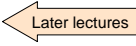
Classes

When you write a Java class, you often have to use the services of other classes; classes that you wrote, and classes written by other people.

There are two main kinds of classes:

- "Container class": a grouping of methods that, typically, have something in common. For example, Java's `Math` class features mathematical functions; The `Random` class features random number generation functions, etc.
- "Definition class": implements the data and operations of some abstraction, e.g. `Turtle`, `Date`, `Car`, ...

Two viewpoints on classes:

- Client view: how to use an existing class  This lecture
- Server view: how to design and implement classes  Later lectures

Outline



- Container classes
 - Example: Math
- Definition class
 - Example: BankAccount
- Creating objects
 - Example: BankAccount
- Object (reference) variables
- Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- The String class
- Packages

The Math class

An example of a container class is Java's `Math` class, which contains a collection of methods that implement all sort of mathematical functions:

\sqrt{x}	Implemented in Java as:	<code>Math.sqrt(x)</code>	} <u>Static methods that take one or more parameters and return a value</u>
$\log_{10} x$		<code>Math.log(x)</code>	
$\text{Min}(x, y)$		<code>Math.min(x, y)</code>	
$ x $		<code>Math.abs(x)</code>	
x^y		<code>Math.pow(x, y)</code>	
...		...	

Syntax for invoking a static method that returns a value:

```
varName = ClassName.methodName(parameters)
```

Example:

```
double x = Math.sqrt(16);  
System.out.println(x);  
// will print 4.0
```

Math class API: <http://java.sun.com/javase/6/docs/api/java/lang/Math.html>

The Math class in action: quadratic equation example

Task: input the coefficients of a quadratic equation and print its roots

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
public class Quadratic {
    public static void main (String[] args) {
        double a, b, c; // ax^2 + bx + c
        double di scri mi nant, x1, x2;

        // Get a, b, c from the user (omitted)

        di scri mi nant = Math.pow(b, 2) - (4 * a * c);

        // Assumes a positive di scri mi nant and a <> 0.
        x1 = ((-1 * b) + Math.sqrt(di scri mi nant)) / (2 * a);
        x2 = ((-1 * b) - Math.sqrt(di scri mi nant)) / (2 * a);

        System.out.println ("x1 = " + x1);
        System.out.println ("x2 = " + x2);
    }
}
```

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 5

Outline

- Container classes
 - Example: Math
- ➔ ■ Definition class
 - Example: BankAccount
- Creating objects
 - Example: BankAccount
- Object (reference) variables
- Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- The String class
- Packages

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 6

Abstractions

Primitive data types:

Fixed,
defined
by the
Java
language

- Byte
- Short
- Int
- Long
- Float
- Double
- Boolean
- char

Abstract data types (ADT):

Java
class
library

- String
- Date
- Set
- ...

Application
specific

- Fraction
- Matrix
- DNA
- ...

Abstractions, in general:

Application
specific

- BankAccount
- Clock
- Friend
- Person
- Turtle
- ...

Abstractions and classes:

- Abstractions are defined as needed, by software architects
- Abstraction definition starts with requirements analysis
- In OOP, abstractions are implemented as classes
- The classes are written by programmers
- The resulting class implementations can be made public to other programmers.

Bank account abstraction (example, described in informal language)

A bank account is characterized by an account number which is assigned to it when a person opens an account

Other things that characterize a bank account are the name of the account owner and the current balance

Things that we want to do with a bank account:

- Show its current balance
- Deposit money
- Withdraw money
- Transfer money from another account
- Some more operations, to be defined later.

BankAccount implementation (first approximation)

According to the abstraction, each bank account is characterized by two things:

- Data: account number, owner name, account balance
- Operations: show balance, deposit, withdraw, ...

BankAccount interface:

```
public class BankAccount {
    // fields
    private long accountNumber; // generated by the program
    private String name;        // supplied when an account is opened
    private double balance;     // supplied when an account is opened
    // Constructor
    public BankAccount(String name, int balance)
    // Methods
    public double getBalance()
    public void deposit(double amount)
    public void withdraw(double amount)
    public void transferTo(double amount, BankAccount targetAccount)
}
```

At this point we take a client view of this class: we don't care how it is implemented and how it was written; All we care about is how to use it.

Outline

- Container classes
 - Example: Math
- Definition class
 - Example: BankAccount
- ➔ ■ Creating objects
 - Example: BankAccount
- Object (reference) variables
- Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- The String class
- Packages

Creating objects

BankAccount interface: (same)

```
public class BankAccount {
    // fields
    private long accountNumber; // generated by the program
    private String name; // supplied when an account is opened
    private double balance; // supplied when an account is opened
    // Constructor
    public BankAccount(String name, int balance)
    // Methods
    public double getBalance()
    public void deposit(double amount)
    public void withdraw(double amount)
    public void transferTo(double amount, BankAccount targetAccount)
}
```

server

- The BankAccount class can be viewed as a template for creating and manipulating bank accounts
- Using it, we can create and manipulate as many BankAccount instances, or objects, as needed by the application.

Object creation example:

```
Public class SomeClass {
    ...
    BankAccount account1 = new BankAccount("Alice", 0);
    BankAccount account2 = new BankAccount("Bob", 100);
    ...
}
```

client

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 11

Creating objects

Syntax:

```
ClassName varName =
new ClassName(parameters);
```

The parameters serve to initialize the object's state
How do we know which parameters are expected?
We consult the class interface (API)

Example:

```
Public class SomeClass {
    ...
    BankAccount account1 = new BankAccount("Alice", 0);
    BankAccount account2 = new BankAccount("Bob", 100);
    ...
}
```

- Now we have two BankAccount objects
- account1, account2 are called "object variables", AKA "reference variables"
- Each of these variables refers to, or points at, a certain object.

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 12

Outline

- Container classes
 - Example: Math
- Definition class
 - Example: BankAccount
- Creating objects
 - Example: BankAccount
- Object (reference) variables
- ➔ ■ Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- The String class
- Packages

Using BankAccount

```
public class BankAccount {
    // fields
    private long accountNumber; // generated by the program
    private String name;        // supplied when an account is opened
    private double balance;     // supplied when an account is opened
    // Constructor
    public BankAccount(String name, int balance)
    // Methods
    public double getBalance()
    public void deposit(double amount)
    public void withdraw(double amount)
    public void transferTo(double amount, BankAccount targetAccount)
}
```

server

```
public class BankMaketRichFast {
    ...
    BankAccount aliceAcc = new BankAccount("Alice", 0);
    BankAccount bobAcc = new BankAccount("Bob", 100);
    // ...
    aliceAcc.deposit(900);
    aliceAcc.transfer(700, bobAcc);
    // Alice's balance = 200 ; Bob's balance = 800
    ...
}
```

client

Using Turtle

```
public class Turtle {  
    // Fields  
    private double alpha = 90; // turtle direction in degrees  
    private boolean tailDown; // True if the tail is down  
    ...  
  
    // Constructor  
    public Turtle()  
  
    // Methods  
    public void moveForward(double units)  
    public void moveBackward(double units)  
    public void turnLeft(int degrees)  
    public void turnRight(int degrees)  
    public void tailDown()  
    public void tailUp()  
    public void hide()  
    public void show()  
    ...  
}
```

server



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

client

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 17

Method invocations

Code example (meaningless ...)

```
Turtle leonardo = new Turtle();  
leonardo.moveForward(100); // calling a void method with one parameter  
leonardo.hide(); // calling a void method without parameters  
  
double y = Math.sqrt(19); // calling a static method that returns a value  
  
BankAccount bobAct = new BankAccount("Bob", 900) // calling a constructor  
// ...  
System.out.println(bobAct.getBalance()); // two method calls, one within the other
```

- Methods may receive parameters, or not
- Methods may be invoked on objects, or not
- Methods may return values, or not.

Using Classes and Objects, Shimon Schocken, IDC Herzliya, www.intro2cs.com

slide 18

Outline

- Container classes
 - Example: Math
- Definition class
 - Example: BankAccount
- Creating objects
 - Example: BankAccount
- Object (reference) variables
- Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- ■ The String class
- Packages

Strings

String = a sequence of characters, e.g. "Los Angeles"

In Java, character strings are represented by a data type (abstraction) which is implemented by a class called `String`

Examples (meaningless):

```
String name = new String("John Cleese");
String title = "Mr."; // Special object construction,
                    // unique to Strings
String salutation = title + name; // "Mr. John Cleese"

int year = 2012;
String line = "See you in London in " + year;
line = line + "If I can afford it."
```

- Every string of characters is represented by an object of the `String` class
- 'b' and "b" are not the same thing!

- Strings can be concatenated using the `+` operator
- The `+` operator is type-sensitive.

```
public class Addition {
    public static void main (String[] args) {
        System.out.println("2 and 3 concatenated: " + 2 + 3);
        System.out.println("24 and 45 added: " + (2 + 3));
    }
}
```

String methods (a small sample out of about 50 of them)

0 1 2 3 4 5 6 7
City: T e l A v i v

```
String s, ci ty = "Tel Avi v";
char c;
int n;
c = ci ty.charAt(0); // 'T'
n = ci ty.length(); // 8
n = ci ty.indexOf('v') // 5
n = ci ty.indexOf("el") // 4
s = ci ty.subStri ng(2, 4); // "i A"
s = ci ty.subStri ng(3); // " Avi v"
s = ci ty.toUpperCase(); // "TEL AVI V"
s = ci ty.repl ace('v', 'g'); // "Tel Agi g"
```

String objects are
immutable: once a `String`
object has been
created,
neither its value nor its
length can be changed

- See <http://www.j2ee.me/javase/6/docs/api/java/lang/String.html>

Outline

- Container classes
 - Example: Math
- Definition class
 - Example: BankAccount
- Creating objects
 - Example: BankAccount
- Object (reference) variables
- Using classes:
 - BankAccount
 - Turtle
- Invoking methods on objects
- The String class
- ➔ ■ Packages

Packages / class libraries

- Classes are often organized in *class libraries*, also called *packages*
- For example, the `Random` class is part of a package called `java.util` and its full name is `java.util.Random`

```
import java.util.Random; // Gives access to Random's services
// import java.util.*; // Gives access to all the classes in this package
public class RandomNumbers {
    // Generate and print two random numbers
    public static void main (String[] args){
        Random rndGenerator = new Random();
        // We can also omit the import command and use a full class name instead:
        // java.util.Random rndGenerator = new java.util.Random();
        int num = rndGenerator.nextInt();
        System.out.println ("A random int: " + num);
        num = rndGenerator.nextInt();
        System.out.print("Another one: " + rndGenerator.nextInt());
    }
}
```

Packages / class libraries

- Java's Standard Class Library: a collection of ~3,000 classes, organized in ~200 packages, included in every Java implementation
- Examples: `java.lang`, `java.net`, `java.io`, ...
- The `java.lang` package is automatically imported into every Java class
- For example, the full name of the `String` class is `java.lang.String`



Aequitas
(Goddess of fair trade)

Related topics (covered in the book):

- Alias assignments (pp. 144-146)
- Wrapper classes (pp. 168-170)