Java Programming
Unit 3

Bits and Pieces:
arrays, loops, packages, access levels, command line arguments

(c) Yakov Fain 2014

Arrays

- Array is a data structure for storing multiple values of the same type.
- You can't just change the size of the array after it's declared.

```
String []  friends = new String [20];  // Declare and instantiate array
friends[0] = "Masha";        //Initializing the first element
friends[1] = "Matilda";      //Initializing the second element
friends[2] = "Rosa";
…
friends[19] = "Natasha";      // Initializing the last element
```

Another way of initializing an array during its declaration:

```
String []  friends = {"Masha", "Matilda", "Rosa","Sharon"};
```

You can create instances of String objects using the **new** operator:

```
String myGF= new String("Oksana");
```

or

```
String myGF= "Oksana";          // see the quiz in the homework
```

Multi-Dimensional Arrays

Masha	**732 546-7864**
Matilda	**718 456-7834**
…	…

Here's a **two-dimensional** array to store the name and the phone number of each girl:

```
String  friends [][] = new String [20][2];

friends[0][0] = "Masha";
friends[0][1] = "732 456-7834";

friends[1][0] = "Matilda";
friends[1][1] = "718 456-7834";
…
```

Wrapper Classes and Autoboxing

Primitive data types have corresponding *wrapper classes* (e.g. int and Integer).
The wrapper are helpful because:

1. They contain useful methods for manipulating their primitive counterparts.
2. In some cases you can use only objects but not primitives (e.g. in data collections).

```
ArrayList myLotteryNumbers = new ArrayList();

myLotteryNumbers.add(new Integer(6));
myLotteryNumbers.add(new Integer(15));
```

You don't need to explicitly create a new instance for every primitive.
Java will do it for you. It's called *autoboxing*:

```
myLotteryNumbers.add(6);
```

And this is called *unboxing*:

```
int luckyNumber= myLotteryNumbers.get(23);
```

For Loops

Java supports several types of loops that are used to repeat the same action multiple times, for example print the names of each girl from the array friends.

```
int  totalElements = friends.length;
int i;

for (i=0; i<totalElements; i++){
   System.out.println("I love " + friends[i]);
}

// some other code goes here
```

```
i++;   // is the same as i=i+1;

i=0;
i++;   // Now i=1;
i--;   // Now i=0;
```

Why i++; is not the same as ++i; ???

Find out the difference on your own.

To exit from a loop use the keywords `break` or `continue`.

A Simple Iteration With **For-Each** Loop

Iterate a collection without the need to know how many elements it has.

When you see a colon read is as “in”.

```
public class ForEachDemo {

    public static void main(String[] args) {
        String []  friends = new String [20];
        friends[0] = "Masha";
        friends[1] = "Matilda";
        friends[2] = "Rosa";
        friends[18] = "Hillary";
        friends[19] = "Natasha";


        for (String girl : friends){

                System.out.println("I love " + girl);
         }

        System.out.println("The iteration is over");
    }
}
```

Java 8 offers a `forEach()` method on collections, which may be a better option in certain cases than for each loops, especially when used with lambdas in multi-processor environment. See code samples here: http://bit.ly/1izYBJh

While Loops

```
// While Loop

int  totalElements = 10;
int i=0;

while (i<totalElements){

   // Do something
    i++;
}
```

```
// Do While Loop

int  totalElements = 10;
int i=0;

do{

   // Do something
    i++;

} while (i<totalElements);
```

The do while loop will be executed at least once.

Walkthrough

1. Create a new project called Loops. Then compile and test the WhileLoopDemo below.

```
public class WhileLoopDemo {
     public static void main(String[] args) {
          String []  friends = new String [20];
          friends[0] = "Masha";
          friends[1] = "Matilda";
          friends[2] = "Rosa";
          friends[18] = "Hillary";
          friends[19] = "Natasha";

          int  totalElements = friends.length;
          int i=0;

          while (i<totalElements){
               if (friends[i]==null){
                 i++;
                  // Go back to check the while expression
                 continue;
               }

               System.out.println("I love " +
                                   friends[i]);
               i++;
          }

       System.out.println(
                  "The iteration is over");
     }
}
```

2. Notice the **==** operator that's used to compare the values of Sting variables.

Note the program prints only those elements that are not `null`. The operator `continue` changes the execution flow and goes back to check the `while` expression.

3. Change the code to exit the loop as soon as the program finds and prints the name Matilda. Use the `break` keyword for this.

(c) Yakov Fain 2014

Debug a Loop in Eclipse

Debug - Lesson5/src/WhileLoopDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug | Servers

WhileLoopDemo [Java Application]
WhileLoopDemo at localhost:1434
Thread [main] (Suspended (breakpoint at line 15 in WhileLoopDemo))
WhileLoopDemo.main(String[]) line: 15
C:\Program Files\Java\jdk1.6.0_19\bin\javaw.exe (Jun 5, 2010 7:59:48

Variables | Breakpoints

Name	Value
args	String[0] (id=
friends	String[20] (id=
totalElements	20
i	0

WhileLoopDemo.java

```
int  totalElements = friends.length;
int i=0;

while (i<totalElements){
        if (friends[i]==null){
                i++;
                continue;
        }

        System.out.println("I love " + friends
            i++;
    }
```

Outline

WhileLoopDemo
main(String[]) : void

Command-Line Arguments

You can pass parameters to the Java program from a command line:

java TestTax 50000 NJ 2

The method `main(String[] args)` of the class `TestTax` receives this data as a String array called `args`. This array will be automatically created by JVM:

```
args[0] = "50000";
args[1] = "NJ";
args[2] = "2";
```

Command line arguments are always passed as Strings. You'll need to convert them to an appropriate data type using a corresponding wrapper class:

```
double grossIncome = Double.parseDouble(args[0]);
```

Packages

A Java project can have hundreds of Java classes and we organize them in *packages* (think file directories). Say, you work for Accounting department of Acme corporation:

```
package com.acme.accounting;
class Tax {
  // the class code goes here
}
```

The file Tax.java must be stored in the corresponding subdirectory:

com
  acme
    accounting

If the `TextTax` program is located in a different directory, you have two options:

1. import the class Tax

```
import com.acme.accounting.Tax;
class TestTax{…  Tax t = new Tax();   ….}
```

2. Use the *fully qualified* notation for accessing the class:

```
com.acme.accountimg.Tax t = new com.acme.accounting.Tax();
```

Data Access Levels

Java classes, methods and member variables can have `public`, `private`, `protected` and package *access levels*, for example:

```
public class Tax {

   private double grossIncome;
   protected String state;
   private int dependents;

   public double calcTax() {
     // do something here
   }
}
```

- `public` – any other class can access variable/method
- `protected` – only subclasses can access variable/method
- `private` - only other members of this class can access variable/method
- Package – only classes located in the same package can access variable/method.

If the access qualifier is missing, the variable/method has the package access level.

The keyword `final`

- If a method declared `final`, this method can't be overridden.

```
static final double convertToCelsius(double far){
            return ((far - 32) * 5 / 9);
}
```

- If a class is declared `final`, you can't extend it (can't subclass it)

```
final class Tax {…};
```

- The value for the `final` variable can be assigned only once

```
static final int BOILING_TEMP = 212; // in Fahrenheit
```

- In exception handling all the catch parameters are implicitly `final`, but you can expliciely state this:

```
try{
  // do something

} catch (final IOException e){
  // handle error here
}
```

Version Control System Git

- Git is a distributed version control system.
- Github.com is a hosting for git repositories. It's free for open source code.
- Bitbucket.com can also host git repositories, but allows private repositories if no more than 5 users access them

Watch this presentation about Git and Github: http://bit.ly/1iDpOKp.

Create a Github account and upload your homeworks there.

Homework

1. Study the lessons 5 and 6 from the textbook and do the assignment from the Try It sections of these lessons.

2. Find put the difference between ++i and i++

3. Read about *immutable* nature of String objects in Java. Learn the difference between creation of strings with “” vs. `new` operator. Will both if-statement below evaluate to true? Use Eclipse debugger to find out.

```
String gfriend=“Masha”;
if (gfriend==“Masha”){ // true or false?}

String gfriend1 = new String (“Natasha”);
String gfriend2 = new String (“Natasha”);

if (gfriend1==gfriend2){ // true or false?}
```

Homework (cont.)

5. Study various types of Java `for` and `while` loops.

6. Modify the program HelloWorld to process one command-line argument to print hello with the name that was passed in the command line. For example, if you'll start this program as

java HelloWorld Mary

it'll print "Hello Mary"

7. Complete the assignments from the Try it section from Lesson 5.

(c) Yakov Fain 2014

Additional Reading

- The Java Virtual Machine by Bill Venners
  http://www.artima.com/insidejvm/ed2/jvm.html

- Eclipse Shortcuts
  http://www.vogella.com/articles/EclipseShortcuts/article.html

- Constructors of the superclass:
  http://www.dzone.com/links/r/a_frequently_asked_question_about_java_constructor.html

- Data Access Levels: http://bit.ly/9nAHFh

- For loops: http://bit.ly/dbrBVr