ITI 1120
Lab # 2

For today’s lab:

- Go to BlackboarLearn and get the material for Lab 2
- Save all the java programs you find there in the C:\work directory.
- We'll be using them later.

Exercise 1 - Overview of a Java Program

- Start Dr. Java
- Open ("load") the file **`Prog1.java`**
  - You should already have saved this file on your hard drive).

Compiling the Java Program

- To compile **`Prog1.java`** with Dr. Java, click on the button “Compile”. This will compile all files listed in the left window.
- Compiler messages appear under the tab “Compiler Output” at the bottom of the window.
  - Shows if the compilation was successful.
  - Otherwise the compiler produces error messages.

Running a Program

- Now that the program is compiled, you can run it
- Click on "Run" (or type F2)
- This will execute the method `main` of the your program.
- In the Interactions zone (see tab at the bottom), you will see the program output
  - You can also click on the tab "Console" to see only program output with any messages generated by Dr. Java

General Organization

- Source file contains a CLASS.
  - We will always have one class per file.
- A CLASS contains one or more METHODS.
- A METHOD contains declarations, statements, and control structures.
  - This is where you will implement your algorithms.
- A PROGRAM must include a class that has a exactly one method called **`main`**
  - We shall see in the second half of the course how many classes can make up a program.
- COMMENTS can be placed anywhere.

Comments

- Comments are for people reading your program.
  - In them you explain your program in English.
  - The compiler completely ignores them.
- In Java
  - Comments may be placed anywhere you like.
  - On any line, everything after `//` (to the end of the line) is a comment.
  - Anything in between `/*` and `*/` is a comment (including multiple lines)
- See `Prog1.java` as for examples

Types of comments

- Single line comment
  - Everything from `//` to the end of the line is the comment

```
some code  // This is a comment
more code
```

- General comment
  - Everything from `/*` to the next occurrence of `*/` is a comment
  - Can be part of a line `code /* comment */ more code`
  - Can be several lines

```
code /* start of comment
more comment
end of comment */ more code
```

Class Definition

- Has these parts:
  - Keyword `class`
    - A keyword is a word that has special meaning in the Java language. Dr. Java highlights these reserved words by colouring them blue.
    - In this case the keyword `class` tells the compiler that you are beginning the definition of a class.
  - A name for the class
    - `Prog1` is the name of a class
  - Methodes
    - An opening `{` <-- this symbol is called a brace or curly bracket
    - One or more method definitions
    - A closing `}`

- Braces are used to enclose lines of code together to form an instruction block.

Identifiers

- The class has the name **`Prog1`**
- In programming, the official term for the name is an "identifier".
  - Identifiers are used to name many things: classes, methods, variables, etc.
- There are rules for identifiers in Java
  - Only use the digits **`0-9`**, letters **`A-Z a-z`**, the characters **`$`** and **`_`**
  - Identifiers cannot start with **`$`** and it is not recommended to start them with **`_`** (underscore)

`main` method definition

- The definition of `main` starts with a line that we will never change for this course:

```
public static void main(String[] args)
```

- `main` is the name of this method; it is a special identifier, like a keyword.
  - The purpose of the `main` method is to tell Java, “when you run the program, start here.”
- After this opening line comes:
  - An opening `{`
  - The "body" of the method - in the example program main's body consists of two statements
  - A closing `}`
- Next week in the lab session, we shall add another method that will be called by `main`.

The `println` and `print` statements

- The simplest forms:
  `System.out.println( "some string" );`
    - Go to the next line
  - `System.out.print ( "some string" );`
    - Stays on the same line, any new printed character or typed in character will follow the message
- A STRING is a collection of characters, contained in double quotes to mark the start and the end of the string.
- Whatever is between the double-quotes is written ("printed") on the console (the screen).
- After the string is printed, the cursor marking the location of where the computer will print next is moved to the start of a new line.
- Note: the quotes are not part of the string.

The “import” Statement

- Indicates to the compiler which libraries (or set of predefined classes/methods) the program uses (or may potentially use).
- In `Prog1.java`, we are interested in all classes (`*`) and input/output methods (io). For example, this import includes `System.out.println`
  - The current version of Java does not require this particular import; it is done automatically
- There can be many “`import`”, usually placed at the start of the file (and always before any of its classes are used).

Syntax - General Features

- Java is "free format".
  - In general, you can have blank lines and space things the way you like.
  - However, there are some restrictions for how to space and place things. You cannot put spaces (or line breaks) in the middle of names or keywords.
  - There are conventions to make programs more readable and understandable by many people (e.g. indentation).
- Java is case-sensitive.
  - **`class`** and **`Class`** are two different words
    - keywords never use capitals
  - This is a common source of bugs
- Java is VERY PARTICULAR about punctuation.
  - If you miss a semicolon or have mismatched brackets or braces or double-quotes, or if you use a single quote (or two) instead of a double quote, you'll get a syntax error.

Some general rules are:

- All brackets must match with a bracket of the same type, opposite direction (open and close pairs)
  - The open-close pairs must fit ("nest") inside each other
    - You can't do this: `( [ ) ]`
- Double quotes must match up ON THE SAME LINE
- All statements end with a `;` (semicolon)
- Braces are normally NOT followed by a semicolon (there are some exceptions in special cases).
- The class name and the file name should be the same (except of course for the `.java` extension on the file name).

Exercise 2 - `Prog2`

- Try the same thing with `Prog2.java`
- What happened?

Prog2

- You will get error messages because there is one mistake in `Prog2.java` (the quote to end the string in the `println` statement is missing).

- This is what syntax error messages look like
  - Where does it say what line the error occurred on?
  - Why does the compiler think there are two errors?
    - Hint: Notice that Dr. Java colours strings red. Note carefully what is coloured red in this program.

- Fix the error, and re-compile
  - When you fix the error, notice the difference in what is coloured red.

Exercise 3 - `Prog3`

- This program illustrates one of the most common errors. Try it!

Exercise 4 - **Prog4**

- This program shows the difference between println and print. Try it!

Exercise 5 - `Prog5`

- Try to compile and run this program.
- What happened?

Exercise 6 - **Prog6** - Correcting Syntax Errors

- Correct all errors in Prog6.java so that it will produce the following output:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This program used to have lots of problems,
but if it prints all the lines on the screen,
you fixed them all.
             *** Hurray! ***
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Reading Input from the Keyboard

- Older versions of Java used a complicated construction to read from the keyboard. Java now comes with a class called **`Scanner`** that simplifies input. You have seen in class how to use Scanner class.
- However, there is no method for reading a character in Scanner class.
- To keep things simple, we provide the Java class **`ITI1120. (provided in this lab)`**
- To use it, include the file **`ITI1120.java`**, in the same directory as your program. Then you can invoke the methods of this class in order to read a value (or several values) from the keyboard.

The methods of the class `ITI1120`

`ITI1120.readInt( )`	: Returns an integer of type `int`
`ITI1120.readDouble( )`	: Returns a real number of type `double`
`ITI1120.readChar( )`	: Returns a character of type `char`
`ITI1120.readBoolean( )`	: Returns a value of type `boolean`
`ITI1120.readDoubleLine( )`	: Returns a array of `double`
`ITI1120.readIntLine( )`	: Returns an array of `int`
`ITI1120.readCharLine( )`	: Returns an array of `char`
`ITI1120.readString( )`	: Returns a string of type `String`

- The value returned by these methods needs to be assigned to a variable of the right type.
- After the invocation of a method, the program will wait for the data to be entered.
- When you input a value from the keyboard and press ENTER, the program stores the value in a variable that you specify, and continues the execution.

Examples of using the `ITI1120 class`

```
int x = ITI1120.readInt( );
```

- If you enter `123` and press ENTER, `x` will be assigned the value `123`.

- The method `readDouble` functions in a similar way.

More on Reading Input from the Keyboard (an alternative with Java 5.0 and Java 6.0)

- Java now comes with a class called `Scanner` that simplifies input.
- How to use a `Scanner`:
  1. Create a new scanner, and assign it's reference to a reference variable `keyboard`.
  2. Each time you want a value from the keyboard, call a method via the reference variable `keyboard`.
- The method that you call depends on which type of value you want for input (see next page).
  - The scanner will read the characters you type, and convert them – if possible – to a value of the type you requested.

Methods in class `Scanner`

`nextInt( ):`	Returns an integer of type `int`.
`nextDouble( ):`	Returns a "real" number of type `double`
`nextBoolean( ):`	Returns a value of `true` or `false` as a value of type `boolean`
`nextLine( ):`	Returns a `String` with the entire remaining contents of the line.

- The returned value of these method has to be assigned to a variable of corresponding type.

- When your program reaches a call to one of these methods, the program will suspend and wait for your input.

- When you enter a value from the keyboard and press ENTER, then the program will read the input and store the value you entered into the variable you specified.

Examples of using `Scanner`

- Initialize a scanner:

```
Scanner keyboard = new Scanner( System.in );
```

```
int x = keyboard.nextInt( );
```

- If you enter `123` and press ENTER, `x` will have the value `123` .

```
boolean b = keyboard.nextBoolean( );
```

- If you enter `true` and press ENTER, `b` will have the `boolean` value `true`.

```
String s = keyboard.nextLine( );
```

- Method `nextLine` puts ALL characters (including spaces) that you type on a line into a String referenced by `s`.

Exercise 7 - Calculate total price

- Write a java program called "TotalBill" that reads the subtotal and the gratuity rate (i.e. tip rate) and then computes and displays the total.
- Here is a sample run:

```
Your program: Enter the subtotal and a gratuity rate:
User: 21.25    15
Your program: The total is 24.4375
```

- Compile and test the program.

Exercise 7 (algorithm)

GIVENS/INPUT: subtotal, gratuity_rate

RESULTS: total (the total of the bill)

HEADER: total<- TotalBill(subtotal, gratuity_rate)

BODY:

```
    Step 1: Read in subtotal and gratuity rate
    Step 2: Compute the total
                gratuity = subtotal * gratuity_rate/100
                total = subtotal + gratuity
    Step 3: Display the total
```

Exercise 8 – Is the number odd?

Write a java program called “OddOrNot” that reads an integer and displays word true if the entered number is odd and otherwise it displays false.

- Here is a sample run:

Your program: Enter an integer.

User: 21

Your program: true

Hint: Recall that an integer is odd if it is NOT divisible by 2, i.e. if the remainder of the division by 2 is NOT equal to zero. Thus use the remainder operator %.

Exercise 9: Capital letter?

Write a program, called “CapitalOrNot” that prompts
The user to enter a character and displays word true
if the entered character is a capital letter and otherwise
prints false.

```
Your program: Enter a character
User: c
Your program: false
```

Together at least

In your assignment, you are asked to put solutions several problems in one .java program. Practice that by placing your solution to Exercise 8 and 9 together in the program called “together”.

Make sure together.java complies and runs both solutions.