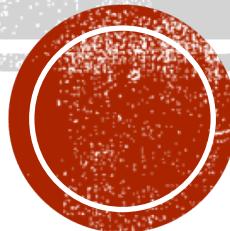


Middlesex
University
London

Lecture 1

Introduction to the module and
Java programming language

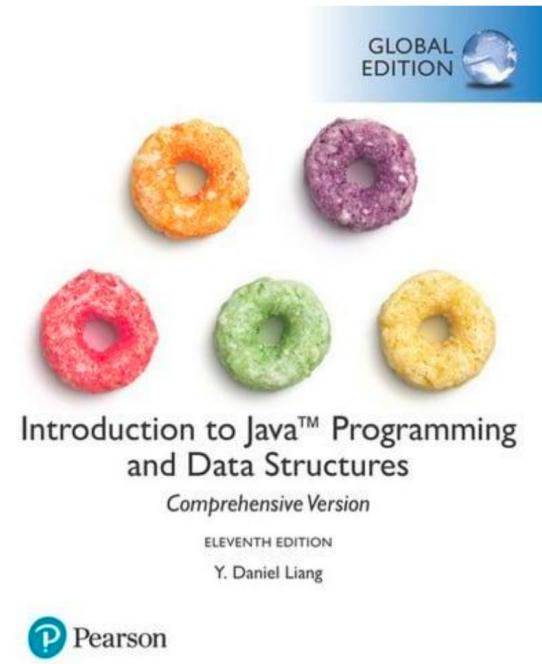


PURPOSE OF THIS MODULE

- This module introduces fundamental computational concepts and programming constructs relevant to understand and use of range of widely used programming languages.
- The module is mainly based on book:
- **“Introduction to Java Programming and Data Structure”**

Eleventh Edition by Y. Daniel Liang

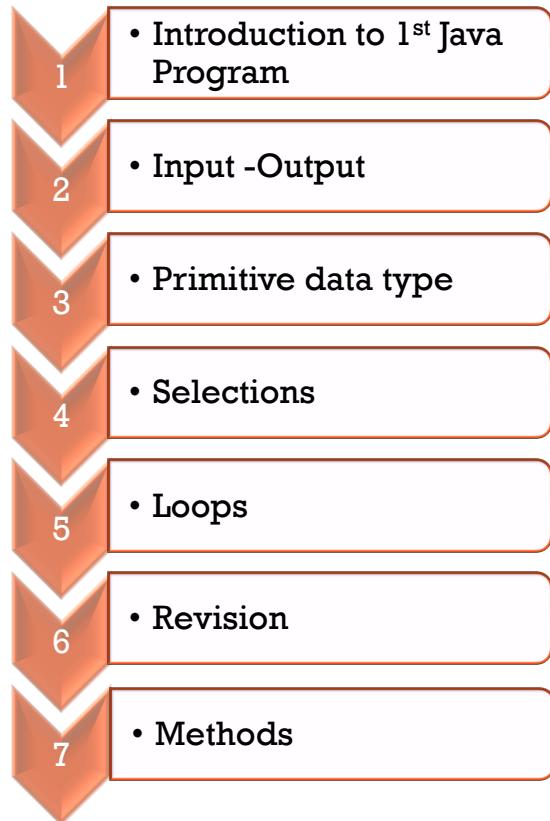
(Available online via university library website)



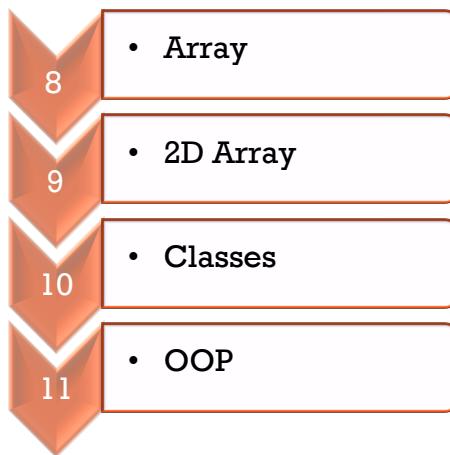
Pearson

STRUCTURE OF THE MODULE

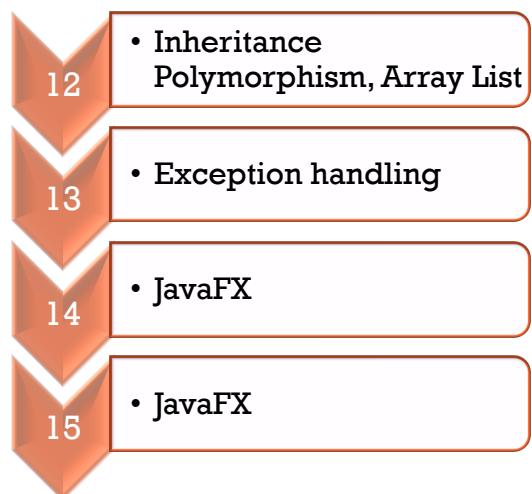
Fundamentals of Programming



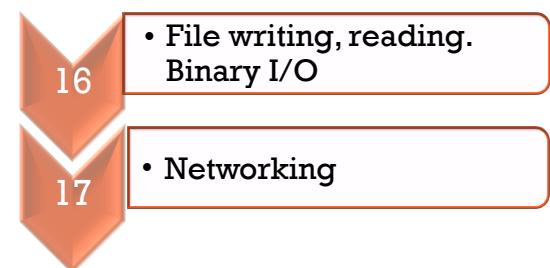
Object Oriented thinking



User Interface and Error Handling



Input Process Storage Output



EXERCISES FOR EACH CHAPTER

Exercises from the book and sobs from the Workbook are prepared for this module and available on MyLearning (often with code examples), but we will use them ***only to practice and improve your programming skills, not to pass the module.***

They are ordered based on difficulty:

- Threshold
- Typical
- Excellent

BEFORE WE START WITH JAVA!

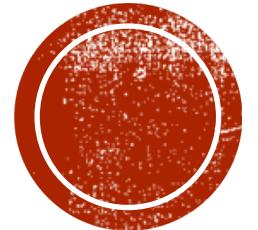
First download and install Java development kit (JDK). It is a package of tools for developing Java based software.

LINK to JDK for this module.

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

On the website choose **Java Standard Edition** (Java SE) the right version for your OS. For most of you it will be the Windows x64 version or macOS x64.

Java SE is the core Java programing edition. It contains the capabilities needed to develop desktop and server applications.



INTRODUCTION TO JAVA



THIS WEEK:

- What is Java
- JDK, API, IDE
- Netbean project
- Java Class
- Java Programs
- Java print, println, printf methods



WHAT IS JAVA?

- One of the most popular Programming languages
- Object Oriented Programming language (OOP)
- Independent from your OS
- Runs on JVM (Java virtual machine)
- Mature language with plenty of development happening



API APPLICATION PROGRAMMING INTERFACES

- Java application programming interface (API) is a list of all classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These pre-written classes provide a tremendous amount of functionality to a programmer.
- A programmer should be aware of these classes and should know how to use them. A complete listing of all classes in
- Java API can be found at Oracle's website:
<http://docs.oracle.com/javase/8/docs/api>

Using Java API may improve performance and shortens program development time.



IDE - INTEGRATED DEVELOPMENT ENVIRONMENT

- IDE is a program which allows and help write the code. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. You do not necessary have to use one.
- The most popular Java IDEs are:
 - Eclipse (<http://www.eclipse.org>)
 - IntelliJ IDEA (<http://www.jetbrains.com>)
 - NetBeans (<http://www.netbeans.org>)



NETBEANS - Creating a new project

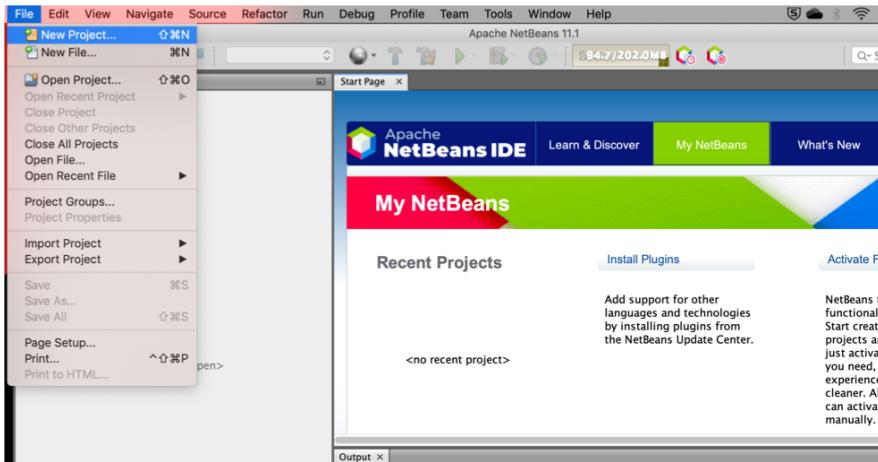


Fig 1. Choose File, New Project

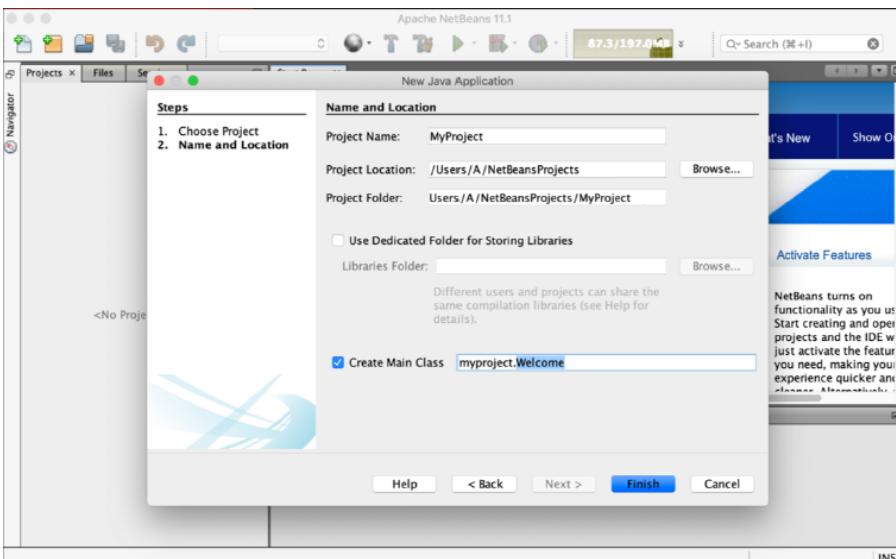


Fig 3. Type name of the project in the Project name, tick the Create main class option, rename the main class field and click Finish

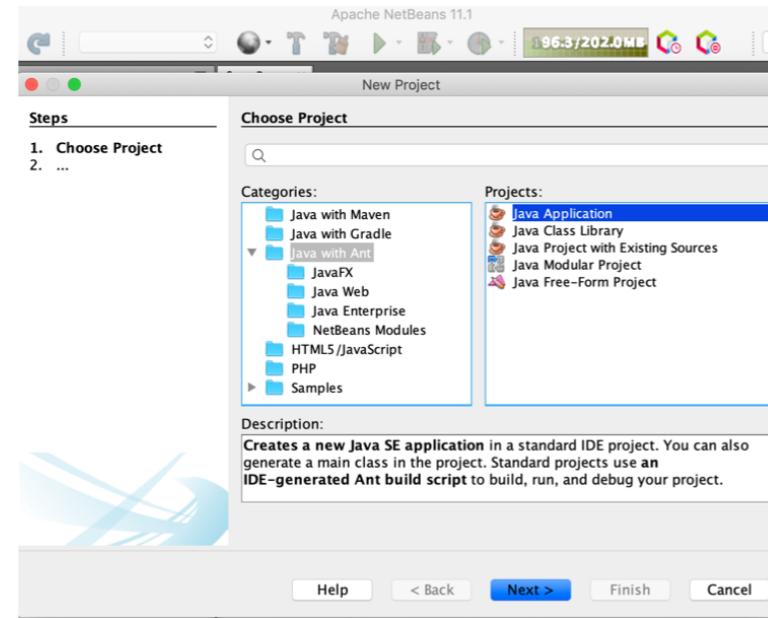


Fig 2. Select Java in the Category section and Java Application in the Project section and then click Next

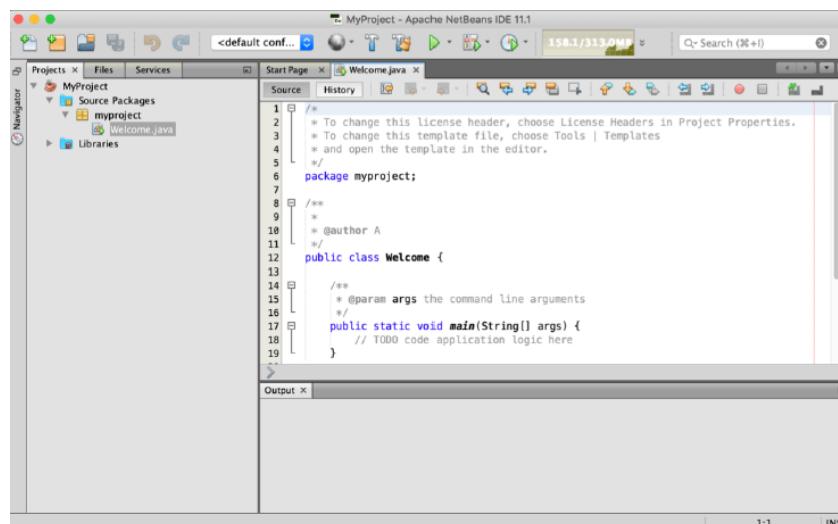
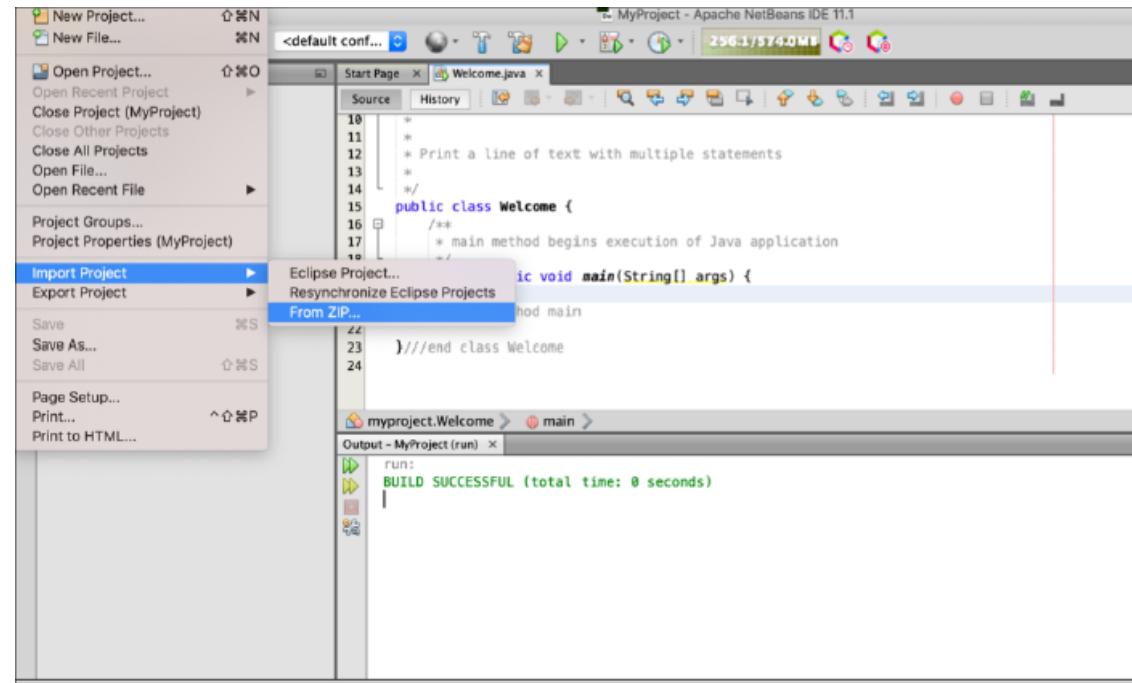
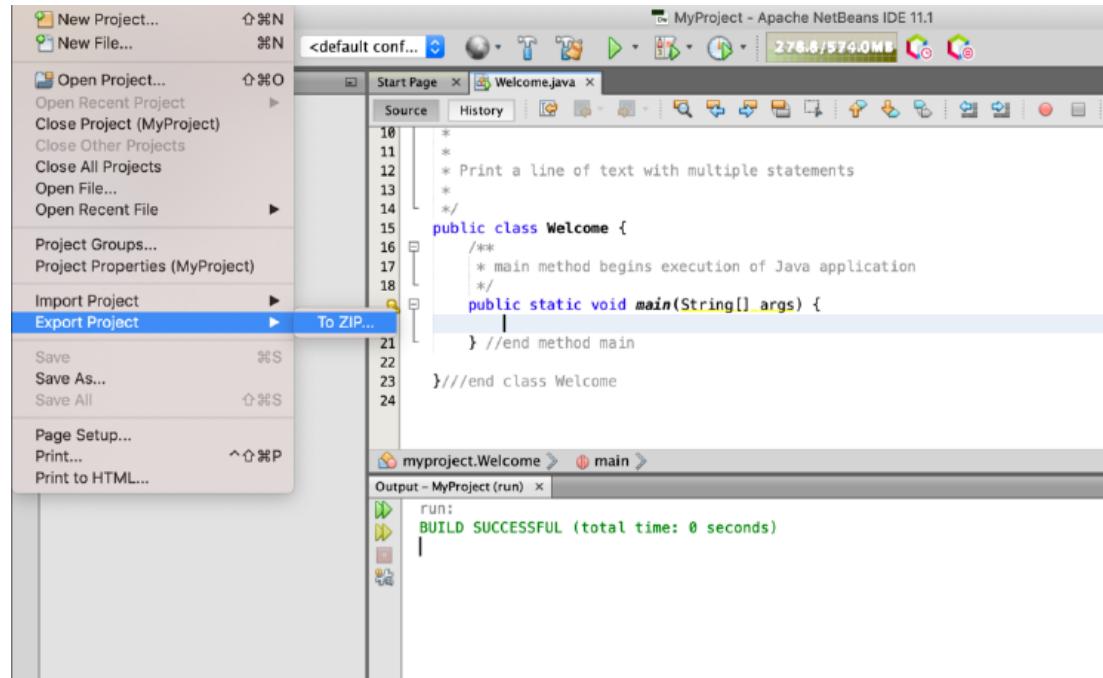


Fig 4. Project name of myproject has been created with a class called Welcome

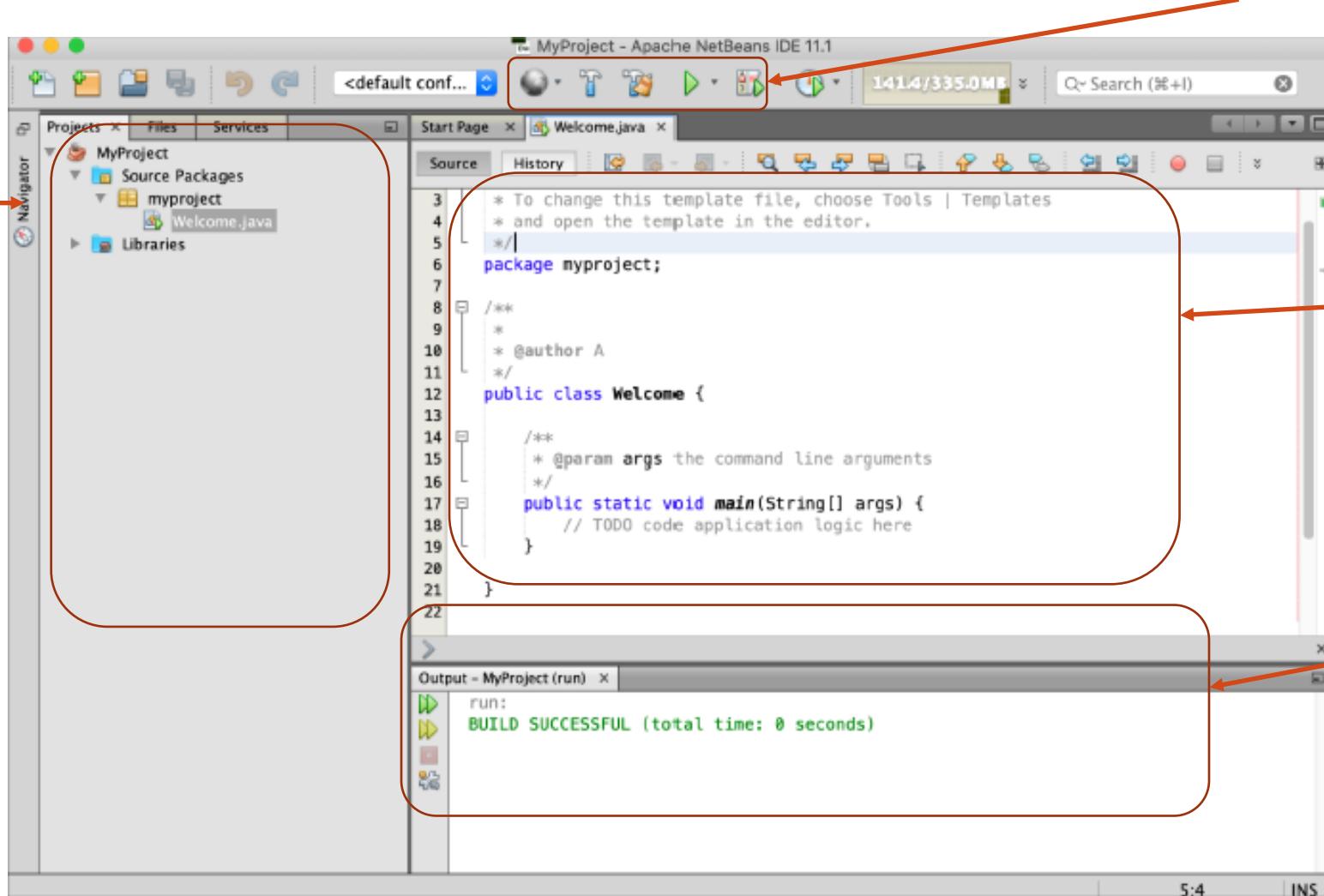
NETBEANS - EXPORTING/IMPORT POROJECT



NETBEANS ENVIRONMENT

Control buttons use to compile and start the program

Project window containing project files



Code editor.
This is the place where the code goes

Console/Terminal
Its where the output of the program is displayed

JAVA CLASS

Name of the package where the class is stored

```
MyProject - Apache NetBeans IDE 11.1
Projects Files Services Start Page Welcome.java
MyProject Source Packages myproject Welcome.java
Libraries

Source History ...
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package myproject;
7
8 /**
9 *
10 * @author A
11 */
12 public class Welcome {
13
14 /**
15 * @param args the command line arguments
16 */
17 public static void main(String[] args) {
18     // TODO code application logic here
19 }
20
21 }
22

Output - MyProject (run) x
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

File name (in this case: **Welcome**) should be the same as the class name
It should start with an uppercase letter

{open and close brackets }
begins and ends the class body

The **main** method is the entry point where program is executed and contains statements

Java is case SEnSiTive



1ST JAVA PROGRAM

The screenshot shows the Apache NetBeans IDE 11.1 interface. The Projects panel on the left shows a project named 'MyProject' with a source package 'myproject' containing a file 'Welcome.java'. The Editor panel displays the code for 'Welcome.java':

```
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 /*
6  * @author A
7  */
8 public class Welcome {
9
10 /**
11 * @param args the command line arguments
12 */
13 public static void main(String[] args) {
14     // TODO code application logic here
15     System.out.println("Hello World!!!");
16 }
17 }
```

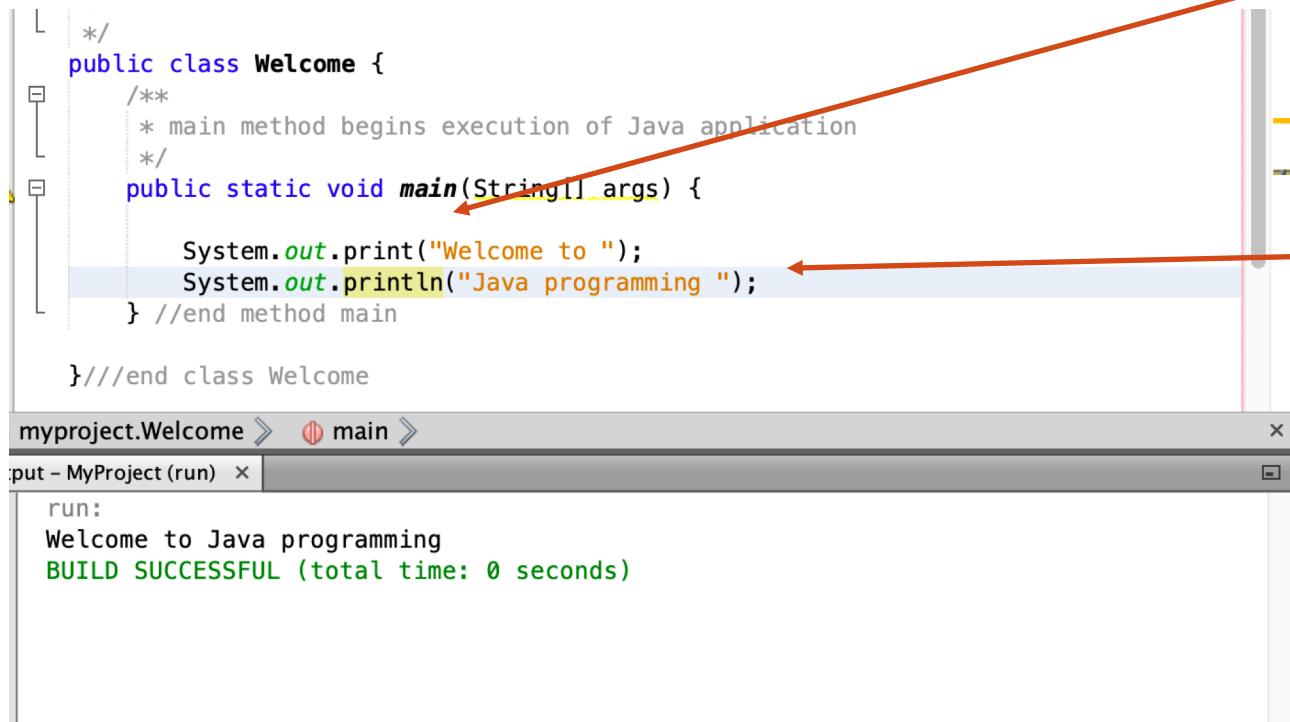
An orange arrow points from the line 'System.out.println("Hello World!!!");' in the code to the output window. The Output panel at the bottom shows the results of running the program:

```
run:
Hello World!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java statement prints
'Hello World!' string of characters
to the console

Output from running of the program

PRINT, PRINTLN METHOD



The screenshot shows a Java IDE interface. The code editor displays a class named 'Welcome' with a main method. The main method contains two statements: `System.out.print("Welcome to ");` and `System.out.println("Java programming ");`. The terminal window below shows the output of running the program: "Welcome to Java programming" followed by "BUILD SUCCESSFUL (total time: 0 seconds)". Red arrows point from the text "print- is used to display the text Without the line separator." to the `print` statement, and from the text "println - is a statement used for displaying the program output in text format using a console/terminal with the line separator." to the `println` statement.

```
/*
public class Welcome {
    /**
     * main method begins execution of Java application
     */
    public static void main(String[] args) {
        System.out.print("Welcome to ");
        System.out.println("Java programming ");
    } //end method main
} //end class Welcome

```

myproject.Welcome > main >

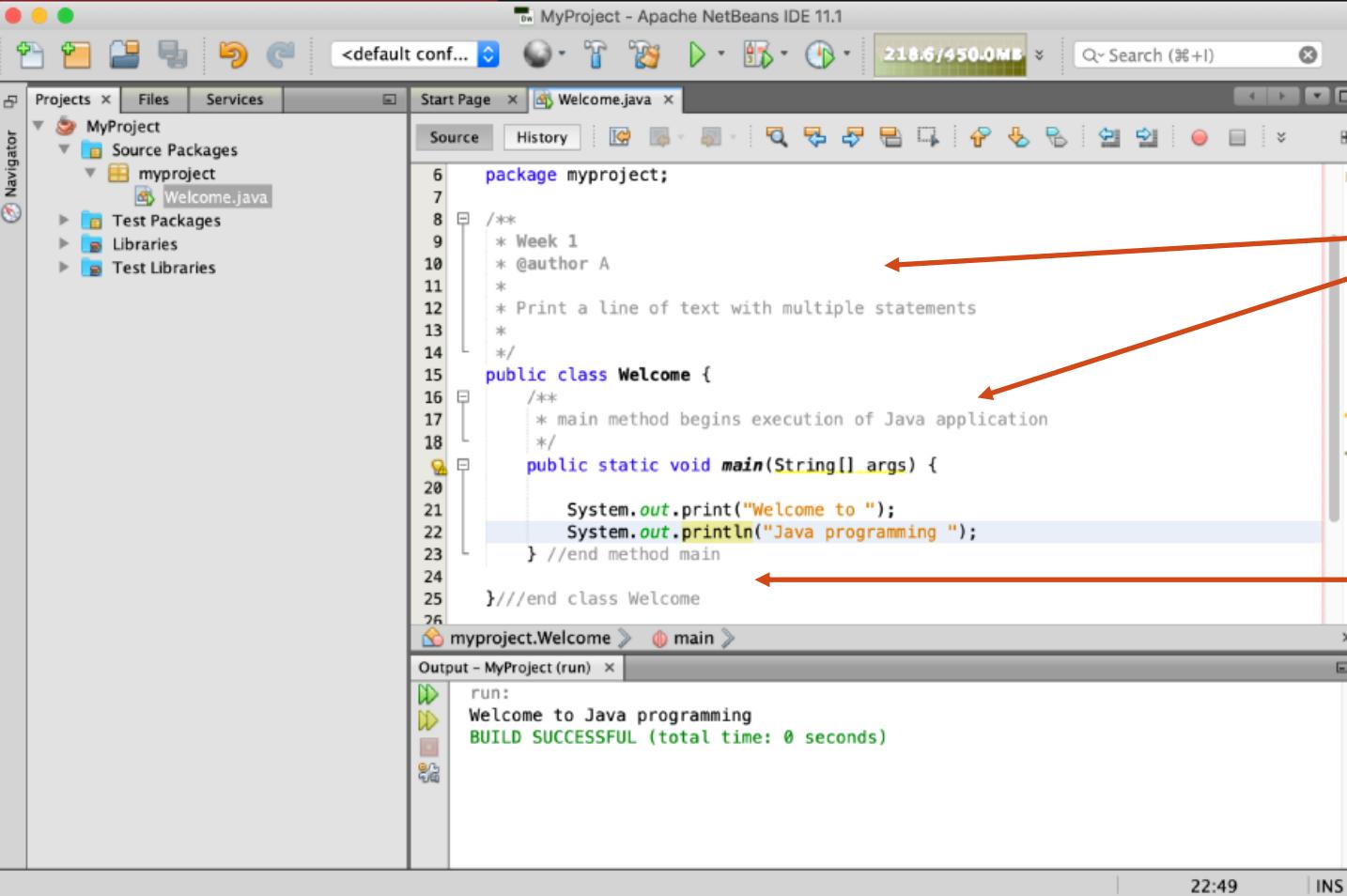
put - MyProject (run) x

```
run:
Welcome to Java programming
BUILD SUCCESSFUL (total time: 0 seconds)
```

print- is used to display the text
Without the line separator.

println - is a statement used for
displaying the program output
in text format using a console/terminal
with the line separator.

JAVA COMMENTS



The screenshot shows the Apache NetBeans IDE 11.1 interface with a Java project named "MyProject". The "Welcome.java" file is open in the editor. The code contains both multi-line and single-line comments.

```
6 package myproject;
7
8 /**
9 * Week 1
10 * @author A
11 *
12 * Print a line of text with multiple statements
13 *
14 */
15 public class Welcome {
16 /**
17 * main method begins execution of Java application
18 */
19 public static void main(String[] args) {
20     System.out.print("Welcome to ");
21     System.out.println("Java programming ");
22 } //end method main
23
24 } //end class Welcome
25
26
```

Annotations in the screenshot:

- A red arrow points from the text "Multi-line comments" to the multi-line comment block starting at line 8.
- A red arrow points from the text "Single line comments" to the single-line comment starting at line 11.

The output window shows the run results:

```
run:
Welcome to Java programming
BUILD SUCCESSFUL (total time: 0 seconds)
```

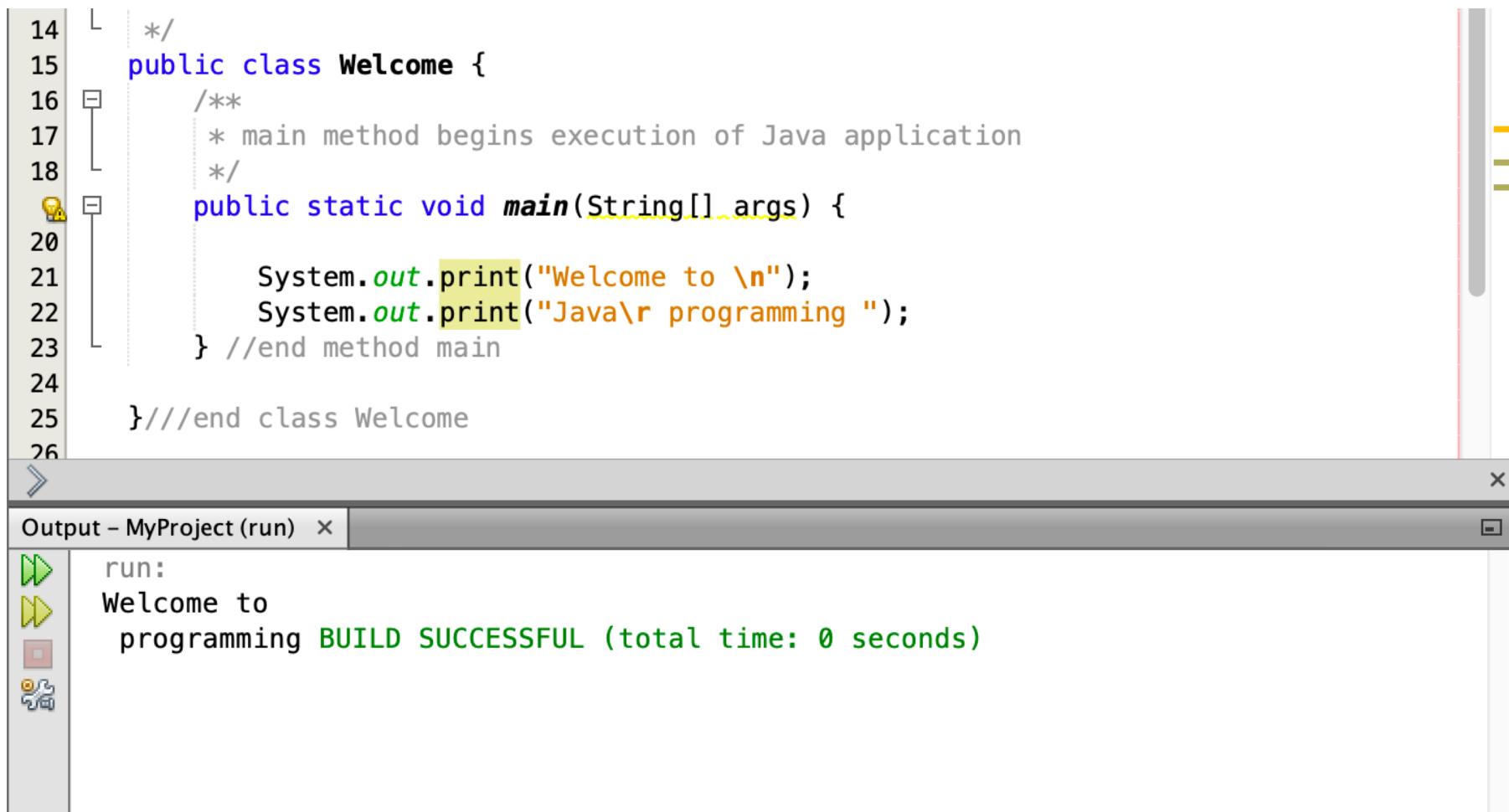


COMMON ESCAPE SEQUENCES

Escape sequence	Description
\n	Newline. (Linefeed)
\t	Tab.
\r	Carriage return.
\\\	Backslash
\\"	Double quote



EXAMPLE OF ESCAPE SEQUENCING



The screenshot shows a Java code editor and an output window. The code editor displays a Java class named Welcome with a main method. The main method prints two lines of text using System.out.print. The output window shows the actual output of the program, which includes escape sequences.

```
14  /*
15   * public class Welcome {
16   *     /**
17   *      * main method begins execution of Java application
18   *      */
19   *     public static void main(String[] args) {
20   *
21   *         System.out.print("Welcome to \n");
22   *         System.out.print("Java\r programming ");
23   *     } //end method main
24   *
25   * } //end class Welcome
26 >
```

Output - MyProject (run) ×

```
run:  
Welcome to  
programming BUILD SUCCESSFUL (total time: 0 seconds)
```

PRINTLN FOR ARITHMETIC OPERATIONS

```
public class Welcome {  
    /**  
     * main method begins execution of Java application  
     */  
    public static void main(String[] args) {  
        System.out.println(5==8);  
        System.out.println(4+7);  
        System.out.println(5%3);  
        System.out.println(2-6);  
    } //end method main  
  
}///end class Welcome
```

put - MyProject (run) ×

```
run:  
false  
11  
2  
-4  
BUILD SUCCESSFUL (total time: 0 seconds)
```



PRINTF METHOD

```
public class Welcome {  
    /**  
     * main method begins execution of Java application  
     */  
    public static void main(String[] args) {  
        System.out.printf("%s%n%s", "Welcome to ", "Java Programming");  
    } //end method main  
  
}///end class Welcome
```

printf - is used to display format data

In this case %s
is a place holder for a string

```
out - MyProject (run) ×  
run:  
Welcome to  
Java ProgrammingBUILD SUCCESSFUL (total time: 0 seconds)
```



PRINTF FORMAT SPECIFIERS:

Place holder	description
%c	Character (Unicode character)
%d	decimal (integer) number (base 10)
%e	exponential floating-point number (float, double)
%f	floating-point number
%i	integer (base 10)
%o	octal number (base 8)
%s	a string of characters
%u	unsigned decimal (integer) number
%x	number in hexadecimal (base 16)
%%	print a percent sign
\%	print a percent sign
%n	the new line character

Can be used later on in the course



KEY THINGS YOU SHOULD UNDERSTAND AFTER WEEK 1

- Compile and run a basic class in NetBeans
- Understand the basic class structure with a main method
- Knowing how to print a line to the console
- Java is a 'strict' language in terms of syntax
- SOBs 1-4



MASTERING YOUR SKILLS

- Make sure that your SOBs for this week are finish
- Read chapter 1 from the Liang book
- Do exercises 1.1 to 1.17 from pages 52, 53 respectivly.

