

Object Oriented Paradigms

College Requirements

CSCR2205

Introduction to Java Programming

Lecture 1-2

What is programming?

- **program**: A set of instructions to be carried out by a computer.
- **program execution**: The act of carrying out the instructions contained in a program.
- **programming language**: A systematic set of rules used to describe computations in a format that is editable by humans.



Compiling/running a program

1. Write it.

- **code** or **source code**: The set of instructions in a program.

2. Compile it.

- **compile**: Translate a program from one language to another.
- **byte code**: The Java compiler converts your code into a format named *byte code* that runs on many computer types.

3. Run (execute) it.

- **output**: The messages printed to the user by a program.



Java Platform

- A Platform is the **hardware** or **software environment** in which a program runs
- Java provide a software Platform
- The java Platform has two components
 - The *Java Virtual Machine* (Java VM)
 - The *Java Application Programming Interface* (Java API)

A Java program

◆ Simple Program

```
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello World From Java");
    }
}
```

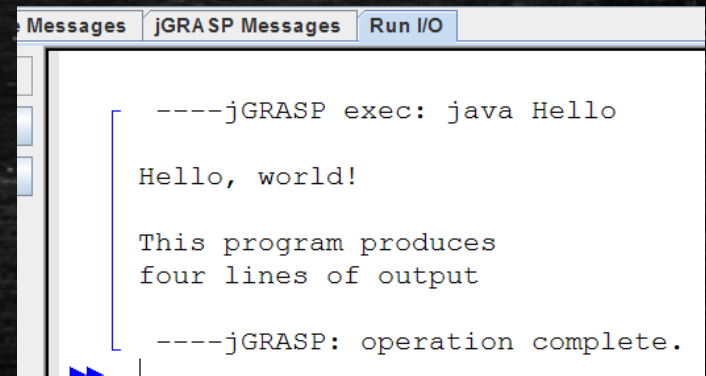

A Java program

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

- **Its output:**

Hello, world!

This program produces
four lines of output



- **console:** Text box into which the program's output is printed.

Comments

- **comment**: A note written in source code by the programmer to describe or clarify the code.
 - Comments are not executed when your program runs.
- Syntax:
 - `// comment text, on one line`
 - or,
 - `/* comment text; may span multiple lines */`
- Examples:

```
// This is a one-line comment.  
/* This is a very long  
   multi-line comment. */
```


Keywords

- **keyword**: An identifier that you cannot use because it already has a reserved meaning in Java.

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular language.
 - Every basic Java statement ends with a semicolon ;
 - The contents of a class or method occur between { and }
- **syntax error (compiler error)**: A problem in the structure of a program that causes the compiler to fail
 - Missing semicolon
 - Too many or too few { } braces
 - Illegal identifier for class name
 - Class and file names do not match

...

Syntax error example

```
1 public class Hello {  
2     pooblic static void main(String[] args) {  
3         System.owt.println("Hello, world!")_  
4     }  
5 }
```

▪ Compiler output:

```
Hello.java:2: <identifier> expected  
    pooblic static void main(String[] args) {  
        ^  
Hello.java:3: ';' expected  
    }  
    ^  
2 errors
```

- The compiler shows the line number where it found the error.
- The error messages can be tough to understand!

Strings

- **string**: A sequence of characters to be printed.
 - Starts and ends with a " quote " character.
 - The quotes do not appear in the output.
 - Examples:
 - `"hello"`
 - `"This is a string. It's very long!"`
- Restrictions:
 - May not span multiple lines.
 - `"This is not
a legal String."`
 - May not contain a " character.
 - `"This is not a "legal" String either."`

Escape sequences

- **escape sequence:** A special sequence of characters used to represent certain special characters in a string.

<code>\t</code>	tab character
<code>\n</code>	new line character
<code>\"</code>	quotation mark character
<code>\\</code>	backslash character

- **Example:**

- `System.out.println("\\hello\nhow\tare \"you\"?\\\\\");`

- **Output:**

```
\hello
how      are "you"?\\
```

Java Types

- Java has two basic types
 - Primitive types
 - Reference Types
- Primitive types
 - integers, floating point numbers, characters, etc
 - Refer to actual values
- Reference types
 - Arrays, Classes, Objects, etc
 - Refer to memory locations (by name, not location)

Primitive Types

Type	Description	Size
Boolean (boolean)	True/false value	1 bit
Byte (byte)	Byte-length integer	1 byte
Short (short)	Short integer	2 bytes
Integer (int)	Integer	4 bytes
Long (long)	Long Integer	8 bytes
Float (float)	Single precision floating point number	4 bytes
Double (double)	Double precision float	8 bytes
Char (char)	Single character	2 bytes

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
 - Like preset stations on a car stereo, or cell phone speed dial:



- **Steps for using a variable:**

- *Declare it* - state its name and type
- *Initialize it* - store a value into it
- *Use it* - print it or use it as part of an expression



variable

```
data type variable [ = value][, variable [ = value] ...] ;
```

- **Example**

```
int a, b, c; // Declares three ints, a, b, and c.  
int a = 10, b = 10; // Example of initialization  
byte B = 22; // initializes a byte type variable B.  
double pi = 3.14159; // declares and assigns a value of PI.  
char a = 'a'; // the char variable a is initialized with value 'a'
```

Interactive programs

interactive program: Reads input from the console.

- While the program runs, it asks the user to type input.
- The input typed by the user is stored in variables in the code.
- Can be tricky; users are unpredictable and misbehave.
- But interactive programs have more interesting behavior.

Scanner

- **Scanner**: An object that can read input from many sources.
 - Communicates with `System.in`
 - Can also read from files (Ch. 6), web sites, databases, ...
- The `Scanner` class is found in the `java.util` package.
`import java.util.*; //so you can use Scanner`
- Constructing a `Scanner` object to read console input:
`Scanner name = new Scanner(System.in);`
- **Example:**
`Scanner console = new Scanner(System.in);`

Scanner methods

Method	Description
<code>nextInt()</code>	reads an <code>int</code> from the user and returns it
<code>nextDouble()</code>	reads a <code>double</code> from the user
<code>next()</code>	reads a one-word <code>String</code> from the user
<code>nextLine()</code>	reads a <i>one-line</i> <code>String</code> from the user

- Each method waits until the user presses Enter.
- The value typed by the user is returned.

```
System.out.print("How old are you? ");  
int age = console.nextInt();  
System.out.println("You typed " + age);
```

- **prompt:** A message telling the user what input to type.

Relational expressions

- `if` statements and `for` loops both use logical tests.

```
for (int i = 1; i <= 10; i++) { ...  
  if (i <= 10) { ...
```

– These are boolean expressions.

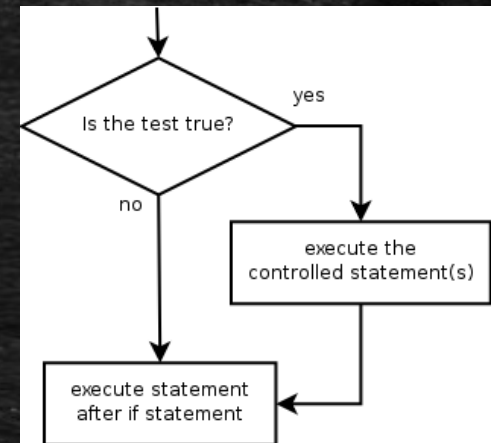
- Tests use *relational operators*:

Operator	Meaning	Example	Value
<code>==</code>	equals	<code>1 + 1 == 2</code>	true
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	true
<code><</code>	less than	<code>10 < 5</code>	false
<code>></code>	greater than	<code>10 > 5</code>	true
<code><=</code>	less than or equal to	<code>126 <= 100</code>	false
<code>>=</code>	greater than or equal to	<code>5.0 >= 5.0</code>	true

The if statement

Executes a block of statements only if a test is true

```
if (test) {  
    statement;  
    ...  
    statement;  
}
```



- **Example:**

```
double gpa = console.nextDouble();  
if (gpa >= 2.0) {  
    System.out.println("Application accepted.");  
}
```

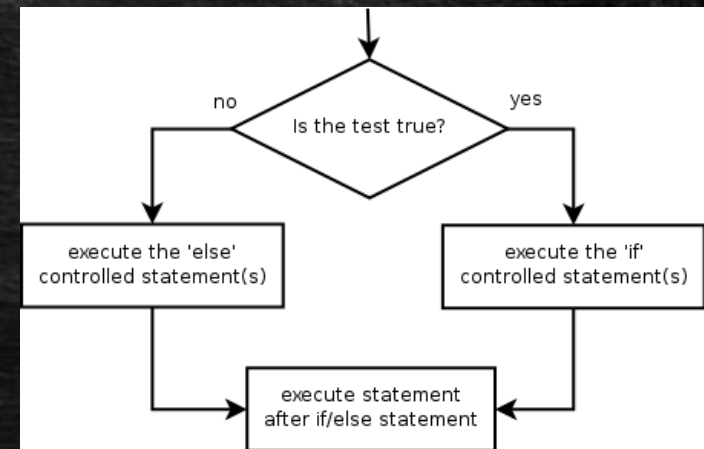

The if/else statement

Executes one block if a test is true, another if false

```
if (test) {  
    statement(s);  
} else {  
    statement(s);  
}
```

▪ Example:

```
double gpa = console.nextDouble();  
if (gpa >= 2.0) {  
    System.out.println("Welcome to Bahir University!");  
} else {  
    System.out.println("Application denied.");  
}
```



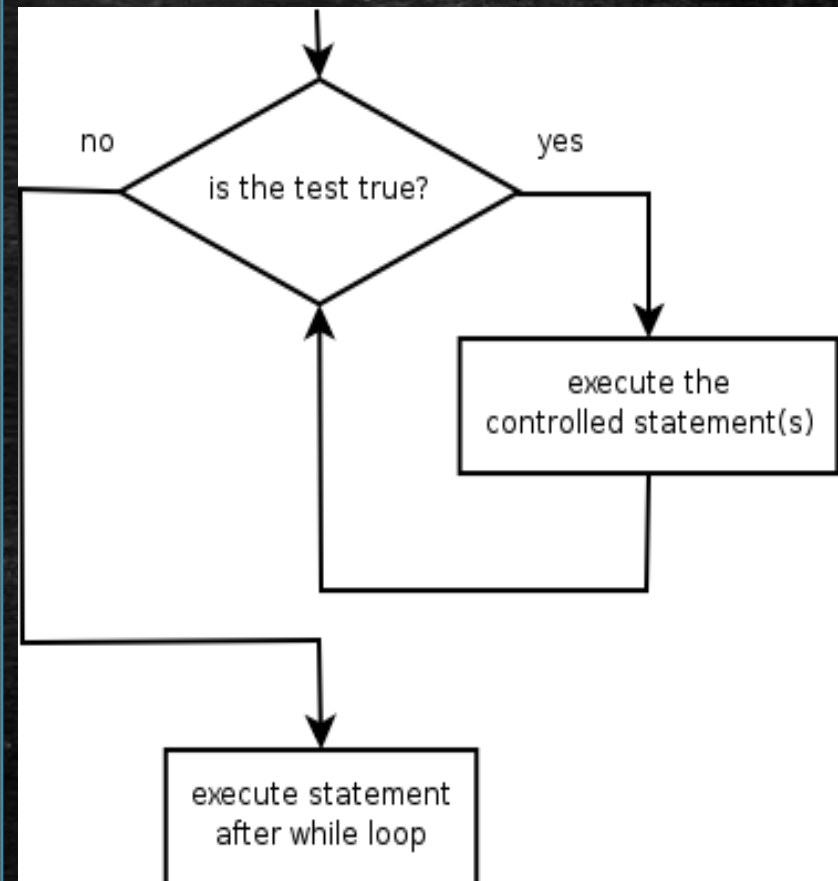
Loop

- There may be a situation when you need to execute a block of code several number of times.

```
•for(initia; test; update)  
  { // Statements }
```

```
initia;  
while(test)  
  { // Statements }
```

```
Initia;  
do {  
    // Statements  
}while(test);
```




Arrays

- **array**: object that stores many values of the same type.
 - **element**: One value in an array.
 - **index**: A 0-based integer to access an element from an array.

<i>index</i>	0	1	2	3	4	5	6	7	8	9
<i>value</i>	12	49	-2	26	5	17	-6	84	72	3

element 0				element 4					element 9
-----------	--	--	--	-----------	--	--	--	--	-----------



Array declaration

```
type[] name = new type[length];
```

```
name[index]           // access
```

```
name[index] = value;   // modify
```

– Example:

```
int[] numbers = new int[10];
```

index 0 1 2 3 4 5 6 7 8 9

<i>value</i>	0	0	0	0	0	0	0	0	0	0
--------------	---	---	---	---	---	---	---	---	---	---

Arrays and for loops

- It is common to use for loops to access array elements.

```
for (int i = 0; i < 8; i++) {  
    System.out.print(numbers[i] + " ");  
}  
System.out.println(); // output: 0 4 11 0 44 0 0 2
```

- Sometimes we assign each element a value in a loop.

```
for (int i = 0; i < 8; i++) {  
    numbers[i] = 2 * i;  
}
```

<i>index</i>	0	1	2	3	4	5	6	7
<i>value</i>	0	2	4	6	8	10	12	14

method

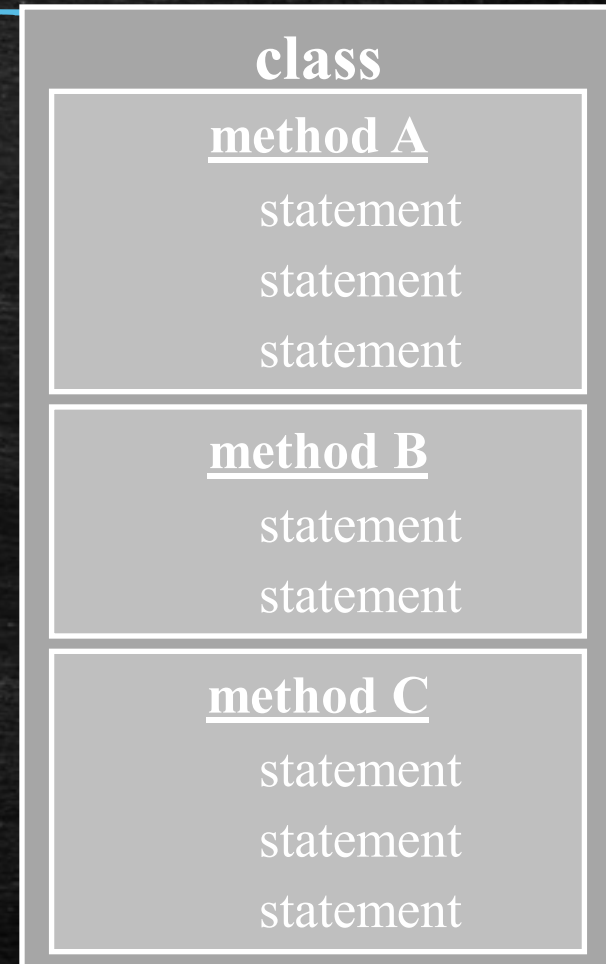
- A Java method is a collection of statements that are grouped together to perform an operation.
- **Creating Method**

```
modifier returnType nameOfMethod (Parameter List)
{
    // method body
}
```

- **Method Calling**

methods

- **Method:** A named group of statements.
 - denotes the *structure* of a program
 - eliminates *redundancy* by code reuse
- **procedural decomposition:**
dividing a problem into methods
- Writing a method is like adding a new command to Java.
 1. **Declare** (write down) the methods.
 2. **Call** (run) the methods.



NOW:

Waiting for your questions and comments

NOW:

Project No 1

Next Lecture

Introduction to OOP