

Introduction to Programming

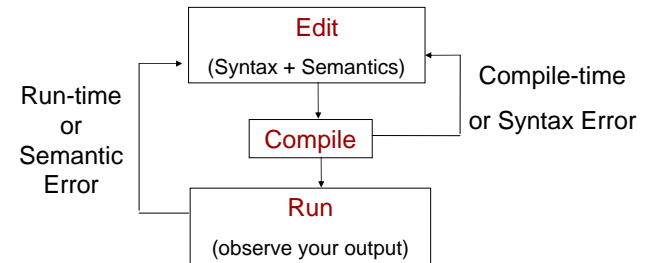
with Java, for Beginners

Process of Programming
Fundamentals:

- Comments & Literals
- Operators
- Primitive Types & Variables
- Expressions/Statements
- Strings

Process of Computer Programming

- To come up with a computation solution



- Philosophy: program in increments

ESE112

2/25

IDE

- Use **I**ntegrated **D**evelopment **E**nvironment
 - A software program that makes it easier to write, compile and run programs
 - We're going to use the free **Dr. Java** IDE
- Setting up Dr Java on your personal PC
 - <http://www.cis.upenn.edu/~palsetia/java/installDrJava.html>

ESE112

3/25

Comments

- Comments are used to make code more understandable to humans
- Java Compiler ignores comments

```
// this is a single line comment

/* this is
 * a multi-line
 * comment
 */
```

ESE112

4/25

Literals

- Literals are the **values** we write in a conventional form whose value is obvious

3 // An **integer** has no decimal point

10.5 // a floating point **double**

'a' // a **character** has single quotes

true // The **boolean** literals are of two types: true, false

"hello world" // A **string** literal

ESE112

5/25

Arithmetic Operators

- + to indicate addition
- - to indicate subtraction
- * to indicate multiplication
- / to indicate division
- % to indicate remainder of a division (integers only)
- parentheses () to indicate the order in which to do things

ESE112

6/25

Relational Operators

- == equal to
 - != not equal to
 - < less than
 - > greater than
 - <= less than equal to
 - >= greater than equal to
- Note: Arithmetic **comparisons** result in a Boolean value of **true** or **false**

ESE112

7/25

Boolean or Logical Operators

- In English, sentences conditions can be formed using "and", "or", and "not"
 - E.g. If there is a test **and** you did not study for it...
 - In Java
 - || -> OR operator
 - true if either operand* is true
 - && -> AND operator
 - true only if both operands are true
 - ! -> NOT operator
 - Is a unary operator – applied to only one operand
 - Reverses the truth value of its operand
- *Operand: a quantity upon which a mathematical operation is performed

ESE112

8/25

Expressions, Operators, Values

- An *expression* has a *value*
- An expression *may* consist of literals and operators
- Given an *expression*, DrJava prints its *value*

Welcome to DrJava

```
> 3
3
> 3 + 5
8
> 'a' == 'A'      // == Equality operator
false
> true && false  // using the logical AND
> true || false // true (using the logical
OR)
```

*Later we'll see that an expression may contain other things
Such as variables, method calls ...*

ESE112

9/25

Values, Types and Expression

- **Value:** Piece of data
23, 10.5, true, 'a'
- **Type:** Kind of data
integer, floating point, boolean (true/false), character
- An **expression** has a value or rather evaluates to a value
23 -> 23
10.5 + 2.0 -> 12.5
3 + 5 * 6 -> 33 -> Precedence Order ?
(3 * 4) / 15 -> 0 -> why zero?
true && false -> false

ESE112

10/25

Types: a very important concept!

- All data values in Java have a *type*
- The type of a value determines:
 - How the value is stored in memory
 - What operations make sense for the value
 - How the value can be *cast* (converted) to related values
- Note: Types are very helpful in catching programming errors

ESE112

11/25

Primitive types

- Values that Java knows how to operate on directly
- We will work with 4 of Java's 8 primitive types
 - Integer (*int*)
-1 42
 - Fractional (*floating point*) number (*double*)
.1 3.14159 2.99792458E8
 - Character (*char*)
'J' '山'
 - Truth value (*boolean*)
true false
- Java's other types are: byte, short, long, float

ESE112

12/25

Storage Space for Numerics

- Numeric types in Java are characterized by their *size*: how much memory they occupy
- Integer types

type	size	range
char	2 bytes	0:65535
int	4 bytes	-2147483648:2147483647

- Floating point types

	size	largest	smallest > 0
float	4 byte	3.4E38	1.4E-45
double	8 bytes	1.7E308	4.9E-324

ESE112

13/25

Another Important Type: String

- A **String** is an **Object**, not a primitive type
 - Java also has objects - cover objects later
- String is composed of zero or more **chars**
- A String is a sequence of characters enclosed by double quotes
 - "Java" "3 Stooges" "富士山"
- + means concatenation for strings
 - "3" + " " + "Stooges" ⇒ "3 Stooges"
- Automatic conversion of numbers to strings
 - 3 + " " + "Stooges" ⇒ "3 Stooges"

ESE112

14/25

Variables

- A **variable** is a named place in memory used to store a value
- Variable must always be associated with *type*
 - It tells the computer how much space to reserve for the variable
 - The value stored can vary over time

ESE112

15/25

Identifiers

- Identifiers are names that you as a coder make up
 - Variable names
 - Also class and method names – next topic to cover
- Variable names
 - *Java Rule*: May consist of alphanumeric characters and the underscore `_` and must start with a *letter*
 - *Style Rule*: Should be a noun that starts with an lowercase letter
 - E.g. `sum`, `average`
- If the name has multiple words, capitalize the start of every word except the first (style rule)
 - E.g. `firstName`, `lastName`

ESE112

16/25

Declaring variables

- All variables must be *declared* before being used
 - With a declaration *statement*
- Declaration statement
 - Specifies the *type* of the variable, followed by descriptive *variable name*, followed by *semicolon*(;)
- Examples:

```
int seats;
double averageHeight;
boolean isFriday;
String houseName;
```

ESE112

17/25

Storing value into Variables

- To store values into variable we use the *assignment* operator i.e. "="
 - *Variable = Expression*; -> assignment statement
- Important
 - Assignment statement must end with a semicolon(;)
 - When a variable is assigned a value, the old value is discarded and totally forgotten
- **Examples:**

```
seats = 150;
averageHeight = (2.1 + 1.74 + 1.58)/3;
isFriday = true;
houseName = "gryffindor";
```

ESE112

18/25

Variable value and type

- The *value* of a variable may be changed:

```
x = 57;
```
- However its *type* may not

```
x = true; // this causes an error, compiler will complain
```

ESE112

19/25

Initializing Variables

- It's good idea to declare and initialize a variable in one statement

```
double milesPerHour = 60.5;
String myName = "Diana Palsetia";
```

ESE112

20/25

Constants

- Variables that don't change
 - Once the program is compiled, they do not change over the execution of the program
- Rules
 - Java Rule: Must have the keyword *final* before the *type*
 - Style Rule: Should have all caps for *variable name*

```
final int NORTH = 0;
final int MILES_PER_GALLON = 32;
```

ESE112

21/25

Integer Division

- `> 10 / 3`
3
- `> (double)10 / 3` // 10 is "cast" to a double
3.3333333333333335
- `> 10 / (double) 3` // 3 is "cast" to a double
3.3333333333333335
- `> (double)(10/3)` // (10/3) is "cast" to a double
3.0
Integer division truncates!

ESE112

22/25

Examples of String creation

```
> String s2 = "hello";
> s2 + " you!"
"hello you!"
> s2 = "The result is " + 100;
> s2
"The result is 100"
```

ESE112

23/25

System.out.println(*String*)

- Command that prints *string* to the output screen
- Can also print literals, and expression values
 - The answer is automatically converted to string
- Prints every time on a new line
- Useful in finding semantic errors in a program

```
System.out.println("hello world");
System.out.println("x")
System.out.print("x = " + x);
```

ESE112

24/25

Recap

- An **Expression**
 - Has a value
 - Consists literals and operators – **FOR NOW!**
- A **Statement** (declaration and assignment)
 - Must end with semicolon(;)
 - Tells or commands the computer to do something
- **Comments** are ignored by the computer
 - They are explanations of your program for human beings to read