

Lecture 12

Java – Variables

Mostafa Elhosseini

Professor at College of Computer Science and
Engineering

Taibah University

<https://youtube.com/drmelhosseini>

| Variables

- How do you save the **intermediate results**, and use them again to produce subsequent ones?
- a normal question to ask **how to store the results of these operations**, in order to use them in other operations, and so on
- **Java** will help you with that. It offers **special "boxes" (containers)** for that purpose, and these boxes are called **variables** - the **name itself** suggests that the **content of these containers can be varied** in (almost) any way.



| Variables

- A **variable** is a **named** location in memory that **stores** a value.
- **Values** may be **numbers, text, images, sounds**, and other types of data.
- The contents of a variable can be **changed anywhere** in the program.
- **Constants**, on the other hand, represent values that **do not change** in the course of a program
- To store a value, you first have to **declare a variable**
 - String message;
- This statement is called a **declaration**, because it **declares that the variable message has the type String**

| Variables

- **Each variable has a type** that determines what **kind** of values it can store.
 - int type can store integers like 1 and -5, and
 - char type can store characters like 'A' and 'z'.
- Some types begin with a **capital** letter and some with **lowercase**
 - There is no such type as **Int** or **string**
- you should use names that indicate what the variables mean:
 - String firstName;
 - String lastName;
 - int hour, minute;

| Variables

Java Primitive Data Types

	TYPE NAME	STORAGE SPACE	APPROXIMATE RANGE
INTEGRALS:	int	4 bytes	+/- 2 billion
	short	2 bytes	+/- 32,767
	long	8 bytes	+/- 9.2 x 10 ¹⁸
	byte	1 byte	-129 to 127
FLOATING-POINT:	float	4 bytes	7-8 digits
	double	8 bytes	16-17 digits
CHARACTER TYPE:	char	2 bytes	65,536
BOOLEAN TYPE:	boolean		true and false

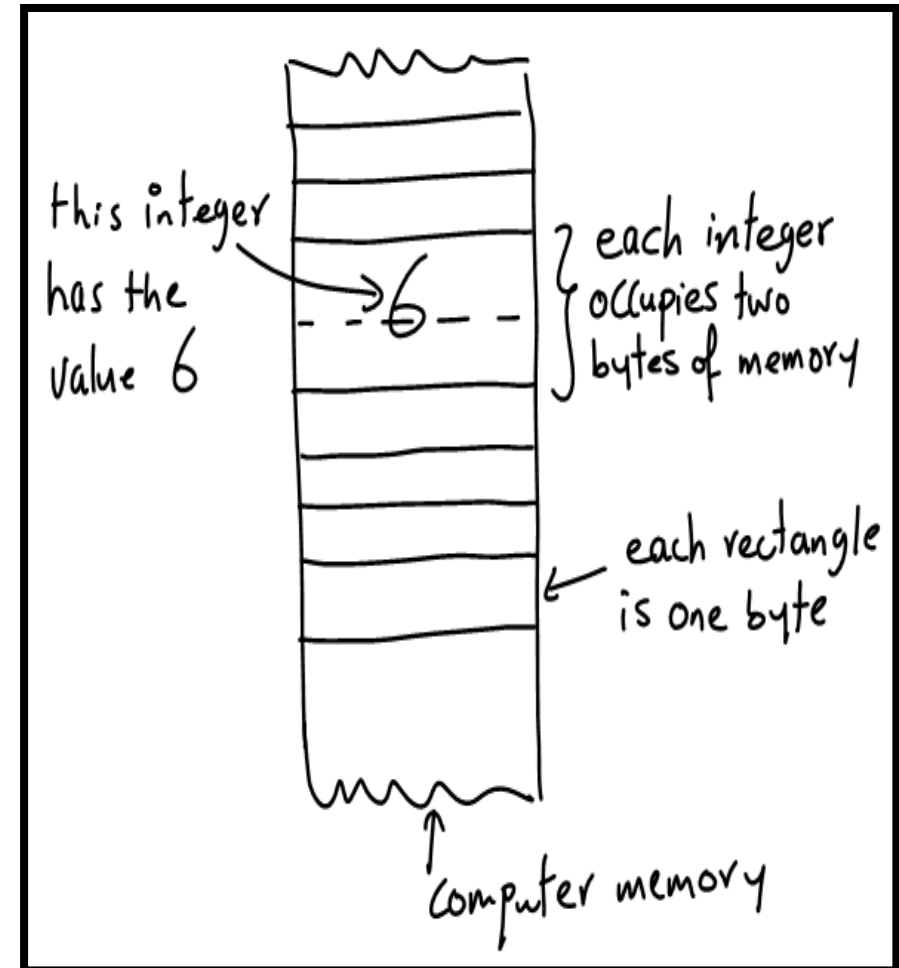
Variable types

Integer %d

Float %f

Character %c

String %s



| Variables

- Note that each **declaration statement** ends with a **semicolon (;)**.
- **Variable names** usually begin with a **lowercase** letter, in contrast to **class** names (like Hello) that start with a **capital** letter
- When a variable name contains more than one word (like firstName), it is conventional to **capitalize the first letter** of each subsequent word.
- Variable names are **case-sensitive**, so firstName is not the same as first name or FirstName - **Identifiers**
- there are about **50 reserved words**, called keywords, that you are not allowed to use as variable names

| Variables

- The following are legal identifiers in Java:
 - personalName
 - PI
 - y_121
 - \$\$128
 - user_address
- The following identifiers are illegal:
 - 1_value
 - User name
 - %%123

| Assigning Variables

- Now that we have **declared** some variables, we can use them to **store** values. We do that with an **assignment statement**:
 - `message = "Hello!";` `// give message the value "Hello!"`
 - `hour = 11;` `// assign the value 11 to hour`
 - `minute = 59;` `// set minute to 59`
- When you declare a variable, you create a named storage location.
- When you **make an assignment** to a variable, you **update** its value
- A variable has to have the same type as the value you assign to it.
 - You **cannot store** a string in minute or an integer in message

| Assigning Variables

- The **equal sign** is in fact an **assignment operator**
- It **assigns the value of its right argument to the left**, while the **right argument** may be an arbitrarily **complex expression** involving literals, operators and already defined variables
 - `var = 1`
- The statement reads: **assign a value of 1 to a variable named var**
- We can say it shorter: **assign 1 to var**
- Some prefer to read such a statement as: **var becomes 1**
- **Expressions** are generally a combination of numbers, variables, and operators. When compiled and executed, they become a single value

| Assigning Variables

- Variables **must be initialized** (assigned for the first time) **before they can be used**.
- You can declare a variable and then assign a value later.
- You can also declare and initialize on the same line
 - `String message = "Hello!";`
 - `int hour = 11;`
 - `int minute = 59;`
- Java uses the `=` symbol for **assignment**
- In Java `a = 7;` is a legal assignment statement, but `7 = a;` is not.
 - The left side of an assignment statement has to be a variable name (storage location).

| Assigning Variables

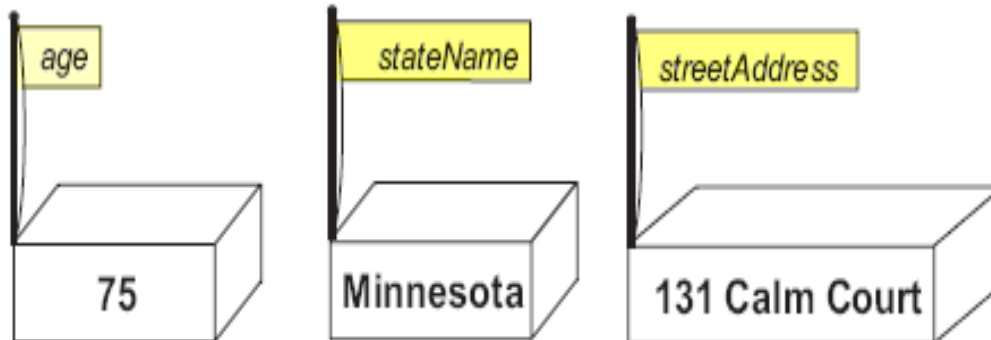
- `int age1;`
- `int age2, age3, age4;`
- `float radius = 1.22;`
- `int valA = 1, val2, val3, val4 = 77;`

| Memory Diagram

- The **variables** in a program and their **current values** make up the **program's state**
- Diagrams like this one that show the state of the program are called **memory diagrams**.
- Each **variable** is represented with a box showing the **name** of the variable on the outside and its **current value inside**

a 3

b 5



| Java String

- Programs frequently use **groups of characters** called **strings**.
- A Java **S**tring is a **class, not a primitive type**.
- The Java String class is part of the **java.lang** library that is directly accessible to any Java program.
- For this reason, you can create objects of the Java String class as if they were variables of a primitive data type.
- Note that because **String** is a **class** it **begins with an uppercase letter**, while all the other primitive data types in the Table have names that start with a lower-case letter

| Java String

- `String uName = "Mansoura - Dakahilia";`
- `String ssn = "12-010-15";`
- The **Java String class** contains several methods to manipulate and convert strings
- The `+` operator has a double life outside the world of mathematics. It can **concatenate** two or more strings. **Glue - Merge**
- Concatenate means to link two things together
 - `String firstName = "Mohamed";`
 - `System.out.println("Everybody loves " + firstName);`

| How to read from User?

- The program prompts the user to **input some data from the console** (most likely using a **keyboard**, although it is also possible to input data using **voice or image**);
- Note you need to assign the result to a variable; this is crucial - missing out this step will cause the entered data to be lost;

| Printing Variables

- You can **display** the current value of a variable by using **print** or **println**
 - `String firstLine = "Hello, again!";`
 - `System.out.println(firstLine);`
- Conveniently, the code for displaying a variable is the same regardless of its type
 - `int hour = 11;`
 - `int minute = 59;`
 - `System.out.print("The current time is ");`
 - `System.out.print(hour);`
 - `System.out.print(":");`
 - `System.out.print(minute);`
 - `System.out.println(".");`

| Formated output

- To print formatted output on the screen
- Format specifier
 - %d : signed decimal integer
- Takes the value on the right of the comma and plugs it into the string on the left – in place of format specifier

| Formatted output

- To print formatted output on the screen
- Format specifier
 - %d : signed decimal integer
 - %s : string: double quotes
 - %c : character single quote
- Takes the value on the right of the comma and plugs it into the string on the left – in place of format specifier

```
System.out.printf("Number of students is %d",45);
```

```
System.out.printf("The grade of %s is %c","Ali",'A');
```

| Mathematical Operators and Routines

- Operators are symbols that represent simple computations.
- Addition operator is +,
- Subtraction is -,
- Multiplication is *, and
- Division is / - Integer Division
- The following program converts a time of day to minutes:

```
int hour = 2;  
int minute = 5;  
System.out.printf("Total minutes = %d",hour*60+minute);
```

| Mathematical Operators and Routines

- Division is / - Integer Division
- In Java, the **default floating-point** type is called **double**, which is **short** for **double-precision**

```
2 public class seko {  
3     public static void main(String[] args)  
4  
5     {  
6         int minute =59;  
7         System.out.println(minute/60);  
8     }  
9 }
```

```
2 public class seko {  
3     public static void main(String[] args)  
4  
5     {  
6         double minute =59;  
7         System.out.println(minute/60);  
8     }  
9 }
```

| Mathematical Operators and Routines

- Very simplest routines use built-in operators
 - $+$, $-$, $*$, $/$, $^$, $\%$
 - Be careful with $/$ on int and long variables
- Static methods in the Math class
 - So you call `Math.cos(...)`, `Math.random()`, etc.
 - Most operate on double precision floating point numbers
 - Simple operations: `Math.pow()`, etc.
 - `pow (xy)`, `sqrt ( $\sqrt{x}$ )`, `log (loge)`, `log10`

| Mathematical Operators and Routines

- Trig functions: `Math.sin()`, etc.
 - `sin`, `cos`, `tan`, `asin`, `acos`, `atan`
 - Args are in radians, not degrees, (see `toDegrees` and `toRadians`)
- Rounding and comparison: `Math.round()`, etc.
 - `round`, `floor`, `ceiling`, `abs`, `min`, `max`
 - Random numbers: `Math.random()`
- `random (Math.random() )`
 - returns from 0 inclusive to 1 exclusive).
 - See `Random` class for more control over randomization.

| References

- Java: How to Program, 9th Edition (Deitel) 9th Edition by Paul Deitel , Harvey M. Deitel
- Head First Java, 2nd Edition 2nd Edition by Kathy Sierra, Bert Bates
- Think Java: How to Think Like a Computer Scientist 1st Edition by Allen B. Downey, Chris Mayfield