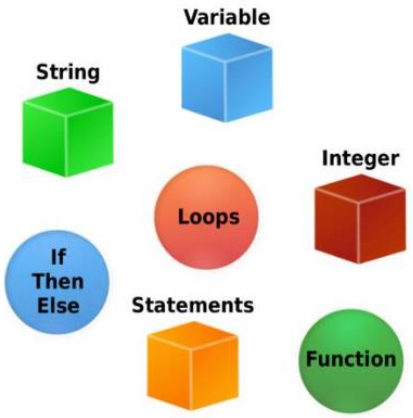
CS2514: Introduction to JAVA



UCC
Coláiste na hOllscoile Corcaigh
University College Cork, Ireland

Primitive Types and Programming Elements

Dr. Krishnendu Guha

Assistant Professor/ Lecturer

School of Computer Science and Information Technology

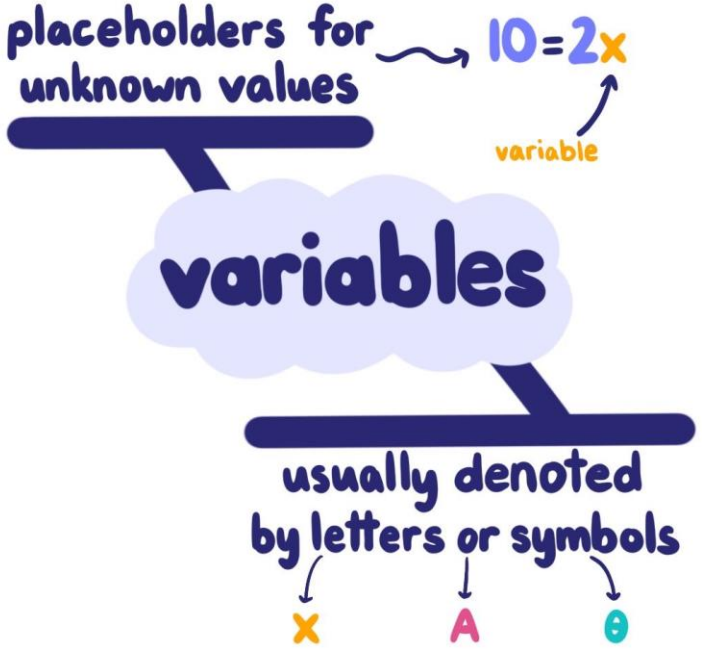
University College Cork

Email: kguha@ucc.ie

Java Variables

Variables are containers for **storing data values**

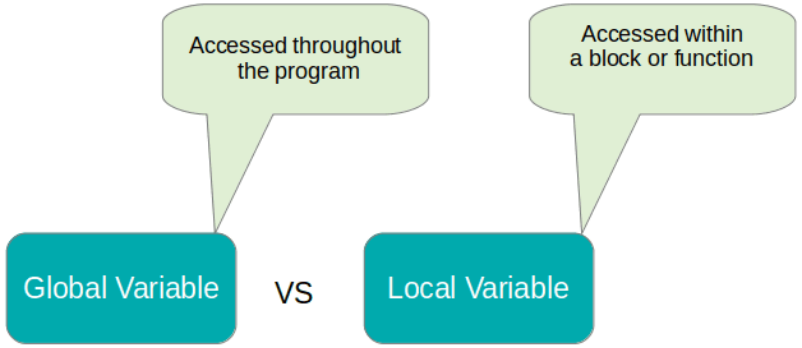
In Java, there are different **types** of variables, for example:
•String - **stores text**, such as “Hi". String values are surrounded by double quotes
•int - **stores integers** (whole numbers), without decimals, such a 123 or -123
•float - **stores floating point numbers**, with decimals, such as 19.99 or -19.99
•char - **stores single characters**, such as 'a' or 'B'. Char values a surrounded by single quotes
•boolean - **stores values with two states**: true or false



Types of Variables in Java

Java has four different kinds of variables:

- local variables - limited
- parameters/arguments - as input to methods
- instance variables (non-static) - individual states of objects
- class variables (static) - belongs to the class




```
#include<iostream>
using namespace std;
// global variable
int global = 5;

// main function
int main()
{
    // local variable with same
    // name as that of global variable
    int global = 2;

    cout << global << endl;
}
```

Global Variable

Local variable

Local Variables

A **variable declared inside the body of the method is called local variable**. They are used **when you need to store** something temporarily during the body of a method.

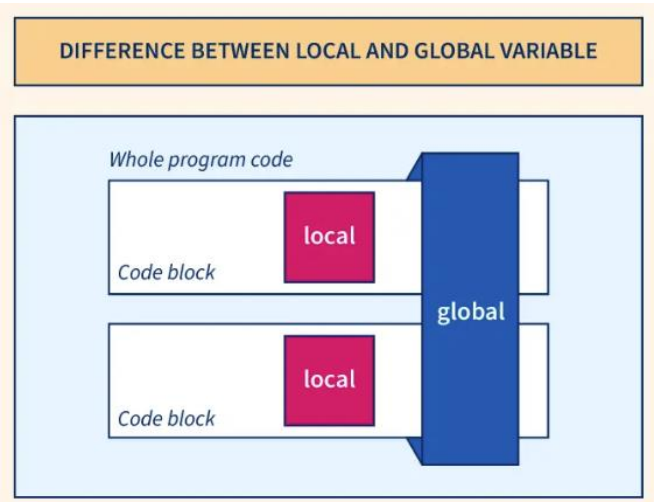
- There is **no specific keyword**.
- It **exists only within the block** that it is declared in (scope).
- You **initialize a local variable by giving it an initial value**.
- You will **get compile-time error if you do not initialize them**!

  (In Python, run-time error if you do not)

```
int n = 3;
```

or declare first and initialize later:

```
int n;
...
n = 3;
```

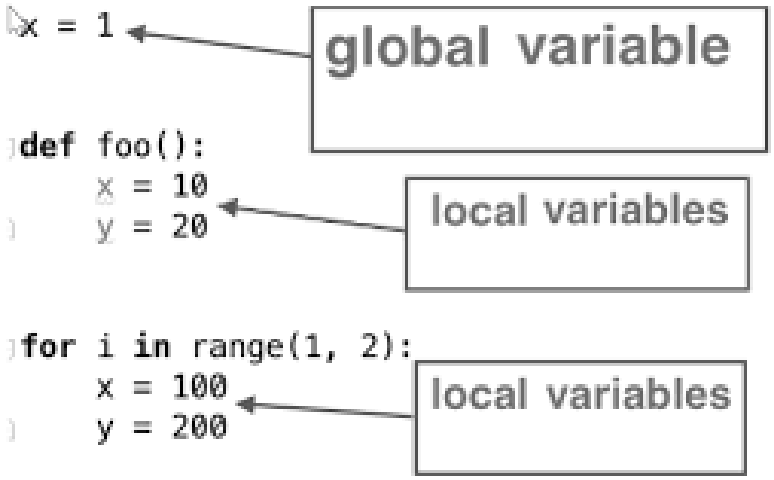


Scope of Local Variables

- In Java, local variables have **block scope**:
- from the declaration of the variable to the end of the block in which it is declared
- if you use the variable's name outside its scope, then **compile-time error**

- In Python, **local variables have function scope**:
- from the **'declaration' of the variable to the end of the function in which it is declared**

if you use the variable's name outside its scope, then treats it as another variable with the same name



Declaring Variables

Declaring (Creating) Variables
To create a variable, we must specify the type and assign it a value:

Syntax
type variableName = ***value***;

Where *type* is one of Java's types (such as int or String), and *variableName* is the name of the variable (such as **x** or **name**)

The **equal sign** is used to assign values to the variable.
To create a variable that should store text, look at the following example:

Example
Create a variable called **name** of type String and assign it the value “**UCC**":

```
String name = “UCC";
System.out.println(name);
```

To create **a variable that should store a number**:
Example
Create a variable called **num** of type int and assign it the value **1**:

```
int num = 15;
System.out.println(num);
```

You can also **declare a variable without assigning any value at the beginning**, and assign the value later:
Example

```
int num;
num = 15;
System.out.println(num);
```

Note that if you **assign a new value to an existing variable, it will overwrite the previous value**:
Example
Change the value of num from 15 to 20:

```
int num = 15;
num = 20; // num is now 20
System.out.println(num);
```

Final Variables

If we **don't want others to overwrite existing values**, use the final keyword (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

Example

```
final int num = 15;
num = 20; // will generate an error: cannot assign a value to a final variable
```

Other Types

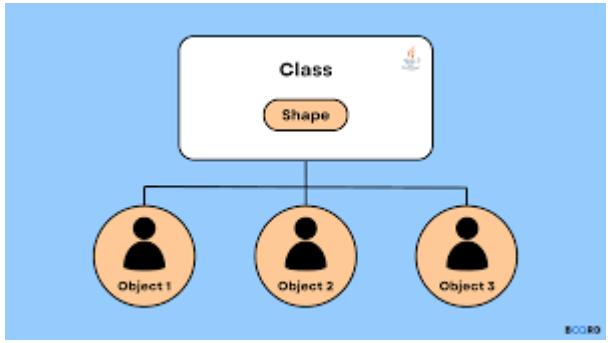
A demonstration of how to declare variables of other types:

Example

```
int num = 1;
float num = 5.9f;
char letter = 'D’;
boolean myBool = true;
String myText = "Hi";
```

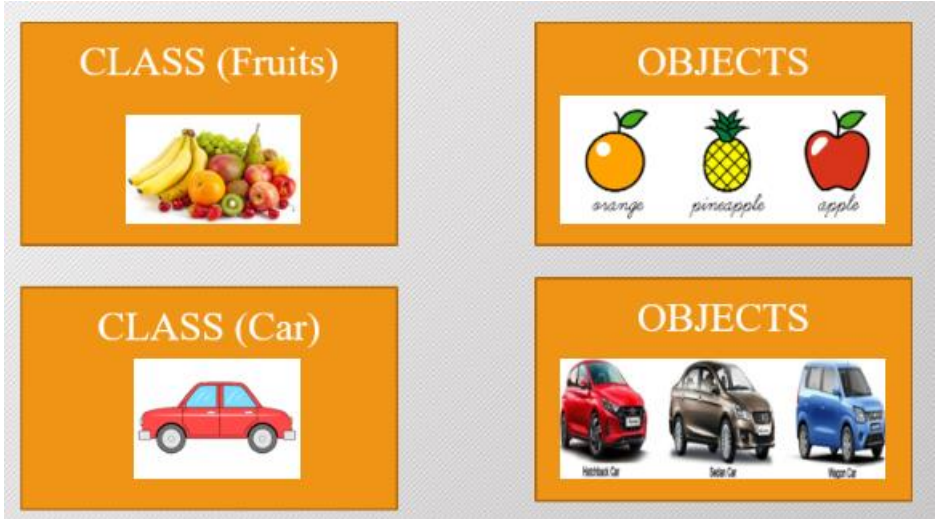
Classes

- Everything in Java is **associated with objects and classes**.
- A **class is a user defined blueprint or prototype** from which **objects are created**.
- It **represents the set of properties or methods that are common to all objects of one type**.
- • E.g. You and I may have different attributes, but we belong to the same class: homo sapiens(human).
- • E.g. System.out belongs to the java.io.PrintStream class



- An **object is an instance of a class.**
- • E.g. Tom is an instance of the Human class.
- • E.g. CS2514 is an instance of the Module class.
- Objects in the same class have the same interface (set of services provided to the clients).

OBJECT	CLASS
Object is an instance ofa class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen. table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newinstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.



Examples

Look around - What do you see?

Things! (books, tables, students, phones)

Individual things!! (my textbook, that table, Ryan, Ryan's phone, etc.)

Describe a particular student:

- Daniel has long blond hair, hazel eyes, is 1.60 tall, is from France, studies computer science, is bored.
- Kate has short brown hair, brown eyes, is 1.80 tall, is from Ireland, studies computer science, is excited.
- We describe them all by specific values of some features: first name, hair length, hair colour, eye colour, height, nationality, degree, emotional state, etc.

```
// Daniel's data
String ageDaniel;
int heightDaniel;
String hairColourDaniel;
// Kate's data
String ageKate;
int heightKate;
String hairColourKate;
// Another Daniel's data
String ageDaniel2;
int heightDaniel2;
String hairColourDaniel2;
```

Instead −→ use classes!

Class Overview

```
public class Student {
	String name;
	String hairColour;
	double height;
	String degree;
	void introduce() {
		System.out.println("Dear stranger, I am " + name + ". Nice to meet you.");
	}
	void changeHairColour(String newHairColour) {
		hairColour = newHairColour;
		System.out.println("I dyed my hair to " + newHairColour);
	}
}
```

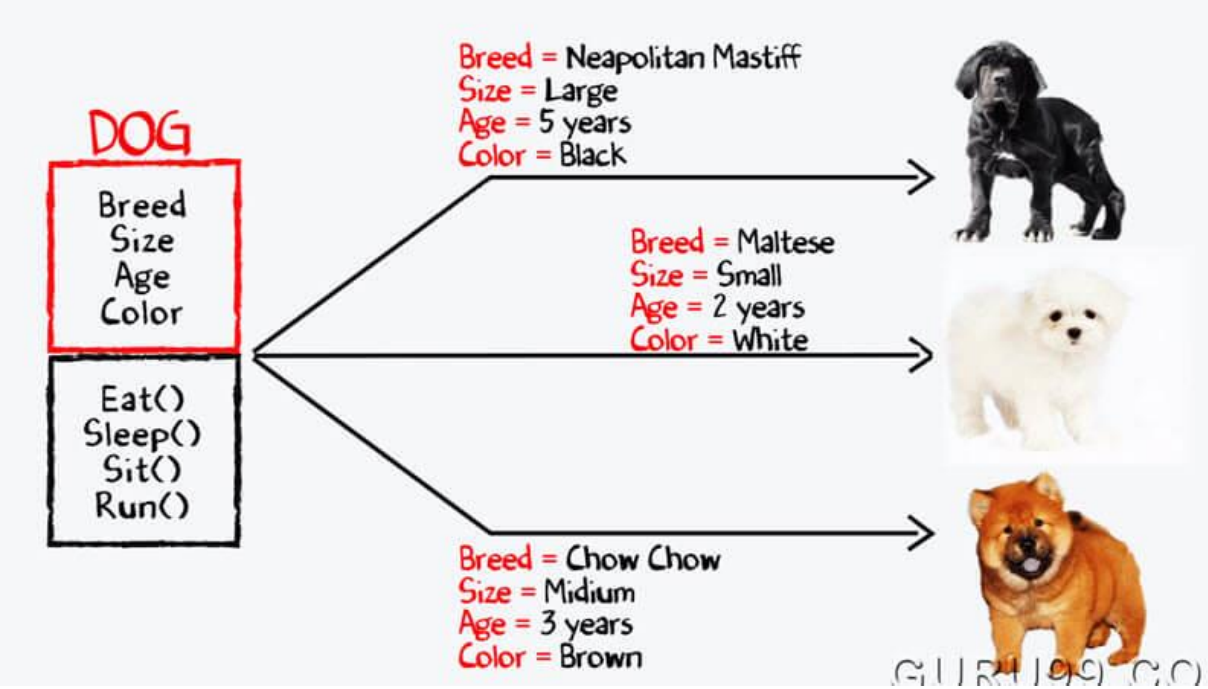
Java OOP Terminology

Class - Category

- properties/states
- functionality/services (examines/alters the state)

Object - An instance of a class

- particular value for each property/state
- same functionality from the class as they all share the same blueprint



What is an Object

The world is a set of things that interact with each other.

Things have:

• state/properties/attributes

• services/functionality/behaviour

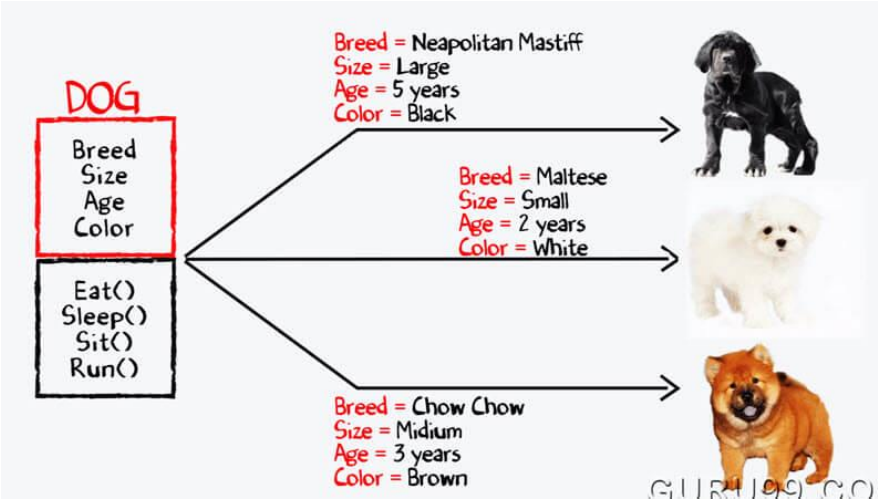
An **object has**

state (what the object knows) and

behavior (what the object can do).

In real world we group things into categories (class) that share some properties, functionalities, and also sub-categories (class hierarchy).

Have 1 class per file.



Comparison

Python - What does the following code do?

```
from random import choice
num = choice( range( 1, 101 ) )
print( num )
```

– **generates a random integer between 1 and 100 inclusive**,
stores it,
prints it

Java, one way to do the same is **to use an object that is an instance of java.util.Random** as there is **no predefined object** for doing this.

We'll have to create one using new, then we can look at its interface and choose which method to use.

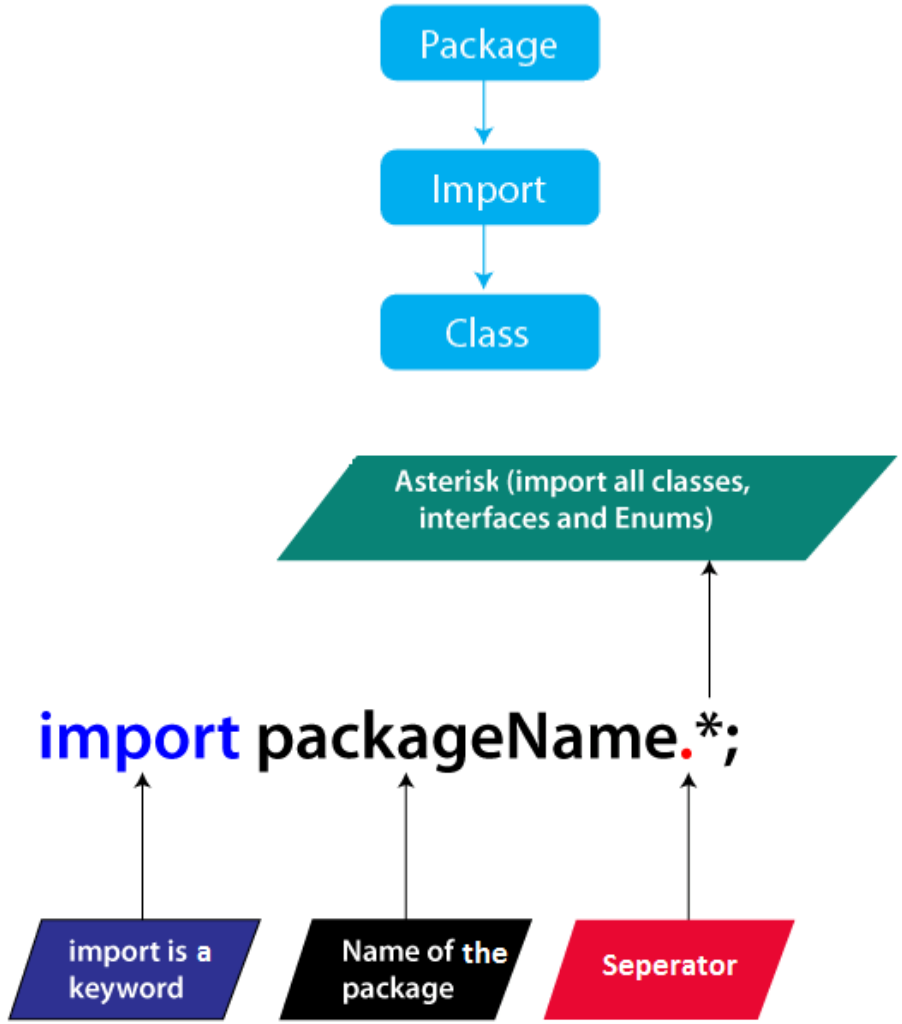
```
class PrintRandom {
	public static void main(String[] args) {
		java.util.Random r = new java.util.Random();
		int num = r.nextInt(101);
		System.out.println(num);
	}
}
```

Import Statements

Java's import statement allows us to abbreviate the names:

```
import java.util.Random;
class PrintRandom {
        public static void main(String[] args) {
                Random randy = new Random();
                int num = randy.nextInt(101);
                System.out.println(num);
        }
}
```

- You don't need import statements for anything from java.lang
- This is why we can use System.out without an import :)



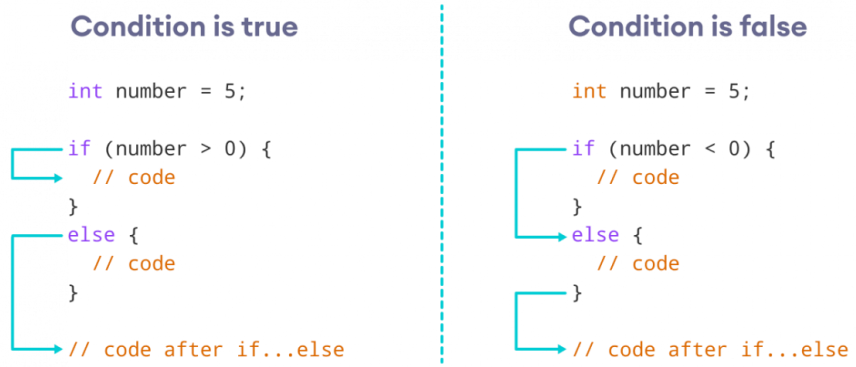
Conditions

The Java if statement:

```
if (condition1) {
statement1
}
else if (condition2) {
        statement2
}
else {
        statement3
}
```

boolean variable: true, false, isOver21

- method with boolean result: exists(), isSent(email)
- operand-comparison operator-operand: age >= 5, speed == 0, firstLetter == 'a'
- boolean expressions combined with logical operator: x < 5 || x > 10, !exists()



If/ else if/ else

Python

```
if x < y:
print('x is smaller than y')
elif x == y:
print('x is equal to y')
else:
print('x is larger than y')
```

Java

```
if (x < y) {
System.out.println("x is smaller than y");
}
else if (x == y) {
System.out.println("x is equal to y");
}
else {
System.out.println("x is larger than y");
}
```

Note

- use the round parentheses for the Boolean expression
- use curly braces to define the scope
- elif does not exist in Java!
- both **Java and Python use comparison operators as: ==, ! =, <,<=, >,>=**
- **Java uses logical operators as: &&, ||, !**
- **Python uses logical operators as: and, or, not**.


Any Questions


Thank you!