CPE207 Object Oriented Programming

Week 2

Declaring a class and creating objects

Dr. Nehad Ramaha,
Computer Engineering Department
Karabük Universities

First program in Java

- In Java, every application begins with a class name, and that class must match the filename.
- Let's create our first Java file, called Main.java, which can be done in any text editor (like Notepad).

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

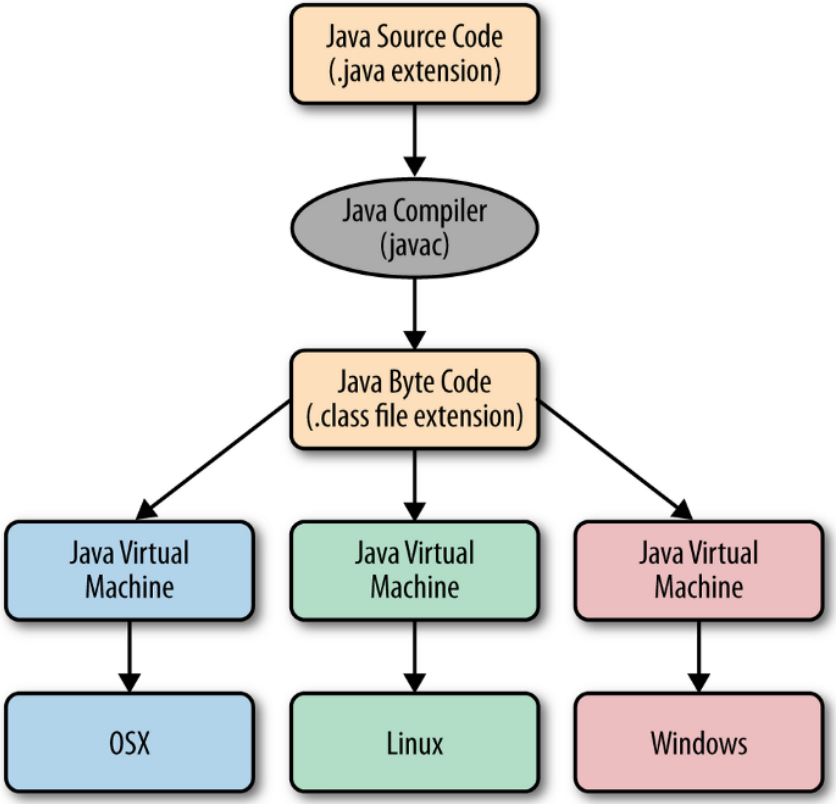
Save the code in Notepad as "Main.java".

- The main() method is required and you will see it in every Java program:

Compile and Run Java program

- Open Command Prompt (cmd.exe), navigate to the directory where you saved your file, and type:
  - javac Main.java
- This will compile your code. If there are no errors in the code, the command prompt will take you to the next line. Now, to run the file type:
  - java Main

The JVM takes the byte code and generates machine code.



Java Editions

- Java Standard Edition (SE) contains the capabilities needed to develop desktop and server applications.
- The Java Enterprise Edition (Java EE) is geared toward developing large-scale, distributed networking applications and web-based applications.
- Java Micro Edition (Java ME) a subset of Java SE. geared toward developing applications for resource-constrained embedded devices, such as:
  - Smart watches
  - MP3 players
  - television set-top boxes
  - smart meters (for monitoring electric energy usage)
  - and more.

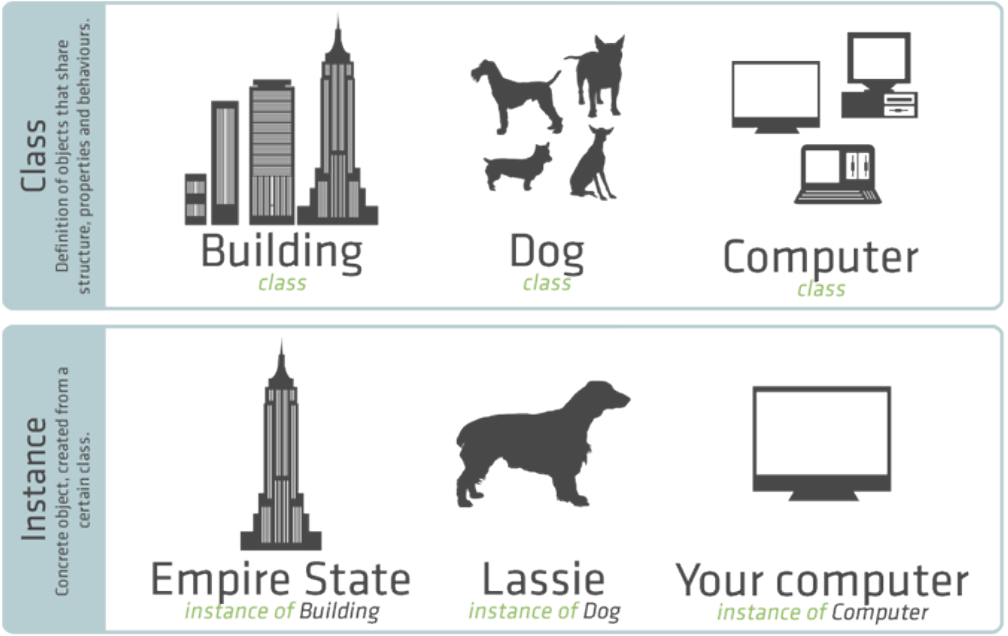
Introduction to Object Technology

- Objects (comes from classes) are *reusable*.
  - Date, time, audio, video, automobile, people objects, etc.
  - Almost any *noun* can be represented as an **object** in terms of
    - *attributes* (e.g., name, color and size) and
    - *behaviors* (e.g., calculating, moving and communicating).

- Object-oriented design approach is much more productive than with earlier popular techniques like "structured programming"
- Object-oriented programs are often **easier to understand, correct and modify.**

Objects and Classes

- **Class** – A *class* is a blueprint or template or set of instructions to build a specific type of object
- **Object**– An object is a component that contains attributes and behaviors needed to make a certain type of data useful.
- **Instance**– An *instance* is a specific object built from a specific class



Class members

- Objects have
  - *attributes* (e.g., name, color and size) and (variables)
  - *behaviors* (e.g., calculating, moving and communicating). (methods)
- A car has *attributes*
  - Color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading).
  - The car's attributes are represented as part of its design in its engineering diagrams.
- Every car maintains its *own* attributes.
- *methods* are used to perform some tasks of the objects.

Instantiation

- Just as someone has to *build* a car from its engineering drawings before you can actually drive a car, you must *build an object* of a class before a program can perform the tasks that the class's methods define.
- An object is then referred to as an instance of its class.

Example

A **class** is a blueprint from which individual objects are created.

```
public class Dog {
    String breed;
    int age;
    String color;

    void bark() {}
    void hunger() {}
    void sleep() {}
}
```

variables

attributes

methods

behaviors

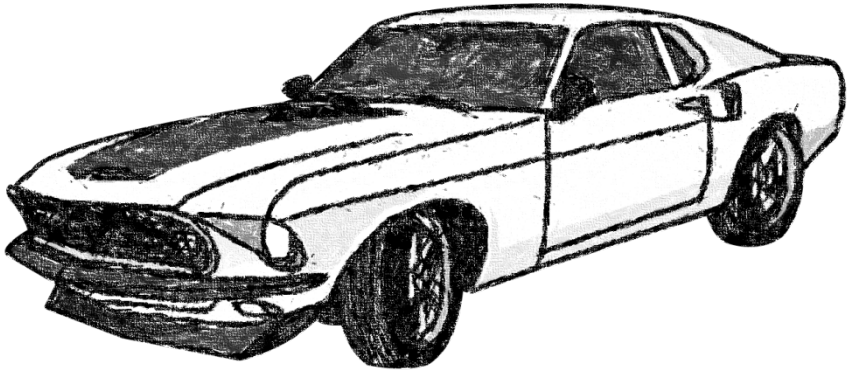
```
public class JavaApplication1 {

    public static void main(String[] args)
    {
        Dog dog = new Dog();

        dog.bark();
    }
}
```

Object: An instance of Dog class

Method call

Another Example: Car class

Attributes
Model
Color
Year
Price



Behaviours
StartEngine
Drive
Stop

This Car class can be *reused* many times to build many cars, you can reuse a class many times to build many objects.

Reuse of existing classes when building new classes and programs saves time and effort.

Need for OOP

Procedure-oriented Programming(POP) vs Object-oriented programming(OOP)

POP

- Main focus is on **"how to get the task done"** i.e. on the procedure or structure of a program
- Large program is divided into functions(methods)
- **C,** VB, FORTRAN, Pascal

OOP

- Main focus is on 'data security'. Hence, only objects are permitted to access the entities of a class.
- Entire program is divided into objects.
- C++, JAVA, C#, Objective-C, phyton

Java Data Types

```
int myNum = 5; // Integer (whole number)
 double myNum = 19.99d; // double
 float myFloatNum = 5.99f; // Floating point number
 char myLetter = 'D'; // Character
 boolean myBool = true; // Boolean
 String myText = "Hello"; // String
```

- Primitive Data Types

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

FORMAT SPECIFIER

- Integer formatted output

```
System.out.printf("Sum is
  %d%n", sum);
```

  - Format specifier %d is a *placeholder* for an `int` value
  - The letter `d` stands for "decimal integer."

FORMAT SPECIFIER	CONVERSION APPLIED
%%	Inserts a % sign
%x %X	Integer hexadecimal
%t %T	Dime and Date
%s %S	String
%n	Inserts a newline character
%o	Octal integer
%f	Decimal floating-point
%e %E	Scientific notation
%g	Causes Formatter to use either %f or %e, whichever is shorter
%h %H	Hash code of the argument
%d	Decimal integer
%c	Character
%b %B	Boolean
%a %A	Floating-point hexadecimal

Java basics

- Conditional statements
  - If-else
  - Switch-case

```java
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

```java
int day = 4;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  case 4:
    System.out.println("Thursday");
    break;
  case 5:
    System.out.println("Friday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
  case 7:
    System.out.println("Sunday");
    break;
}
// Outputs "Thursday" (day 4)
```

Java basics

- Loops
  - For
  - While
  - Do-while
  - For-Each Loop(with arrays)
- Arrays
  - Int[], float[], char[]…

```java
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

```java
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

```java
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

```java
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Why not using Procedure-oriented Programming (POP)? Why need for OOP?

- Task: **computing average score for students.**
  - We put students name, midterm and final exam
  - Create a method to compute average score of a student.
    - `float CalculateScore();`
  - Create another method to show student all info
    - `void ShowStudentInfoAndScore();`

Let's move to IDE

Need for OOP

- There is no need for OOP concepts at all. Almost all the software in the world can be achieved using procedural (POP).

- But OOP allows the programmer to think in terms of objects and classes that makes it
  - much easier to understand the code
  - easy to code and modify it
  - Reuse and existing code
  - Much easier to hide data.

- **It makes software a set of interacting Objects.**

Writing a program using Object-Oriented Programming (OOP)

- Task: Same task
  - We first create a Student class
    - put students name, midterm and final exam
  - Create a student object from the class
    - float CalculateScore();
    - using student object
  - Create a method to show all info
    - void ShowStudentInfoAndScore()

Declaring a class

```
public class Dog {
    String breed;
    int age;
    String color;
    void bark() {…}
    void hunger() {…}
    void sleep() {…}
}
```

- Every Java program consists of at least one class that you define.
- `class` keyword introduces a class declaration and is immediately followed by the class name.
- A `public` class must be placed in a file that has a filename of the form *ClassName*`.java`, so class `Dog` is stored in the file `Dog.java`.
- A left brace, `{`, begins the body of class declaration.
- A corresponding right brace, `}`, must end each class declaration.

Using java built-in objects Scanner class: reading Keyboard

- An object *must* be declared with a name and a type before they can be used.
- Declaration statement

```
Scanner input = new Scanner(System.in);
```

- Scanner Class
  - Enables a program to read data for use in a program.
  - Data can come from many sources, such as the user at the keyboard or a file on disk.

Using java built-in objects Scanner class: reading Keyboard

```
public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Please enter first number: ");
        int numberOne = input.nextInt();

        System.out.println("Please enter second number: ");
        int numberTwo = input.nextInt();

        int result = numberOne + numberTwo;
        System.out.printf(“ the result is %d %n“, result);
    }
```

Code Conventions: Method Names

Methods can either return something or return void. If a method returns something, then its name should explain what it returns, for example:

- void get() BAD!
- boolean isValid(String name); **GOOD!**
- String content(); **GOOD!**
- int ageOf(File file); **GOOD!**

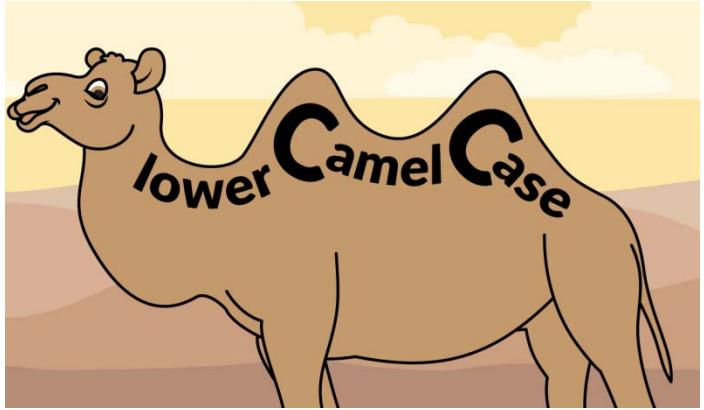
Code Conventions

If it returns void, then its name should explain what it does. For example:

Method Names

- void saveToLocation(File file);
- void process(Work work);
- void append(File file, String line);

•Variable Names
•Eg: name, age. mobileNumber



Thanks ☺

Exercise for Lab (Week 2)

- Create an Employee class includes
  - Attributes
    - name (string)
    - socialSecurityNumber (int)
    - Wage (float)
    - workingHours (int)
  - Behaviours
    - displayInfo()
      - Print name with socialSecurityNumber;
    - displaySalary()
      - Print salary (= wage * workingHours)
- Create an employee object and initialize attributes.
- Call displayInfo() and displaySalary() methods for employee object.
- Create another employee, and this time get attributes from the keyboard