CSC 3215: Object Oriented Programming - 1 (JAVA)

Data Types in Java

Dr. Kamruddin Nur

Associate Professor
Dept. of Computer Science, AIUB
kamruddin@aiub.edu

January, 2018

Contents

Data Types in Java

Data types: A data type is a **set of values** and a **set of operations** defined on those values.

Java has -

1. Primitive data-types
   - Arithmetic data types
     - Integral types:
       - `byte`
       - `short`
       - `int`
       - `long`
     - Floating point types:
       - `float`
       - `double`
   - Boolean data type
     - `boolean`
   - Character data type
     - `char`

Data Types in Java *(Cont'd)*

2. String data type
   - `String` (special support by `java.lang.String` class)
3. User-defined data types
   - Java let's us create user-defined data types using Java's class mechanism
   - User defined data types are those that a programmer himself defines as classes
   - e.g. `Account.java`, `Person.java`, `Car.java` etc.

Note: The default value of String or any object is `null`

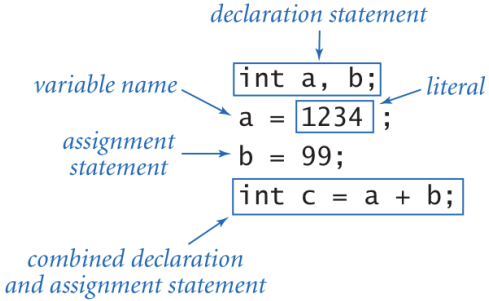
Primitive Data Types

Table 1: Primitive Data Types in Java

Type	Size	Default	Example Literals
boolean	1 bit	false	true, false
char	16 bits (unicode)	\u0000	'a', '\u0041', '\'
byte	8 bits	0	-128,0,127
short	16 bits	0	–1, 0, 32000
int	32 bits	0	-1, 0, 1, 150000
long	64 bits	0L	-1L, 0L, 1L, 100L
float	32 bits	0.0f	-1000.50f, 0f, 1000.50f
double	64 bits	0.0d	-100000.50fd, 0d, 100000.50d

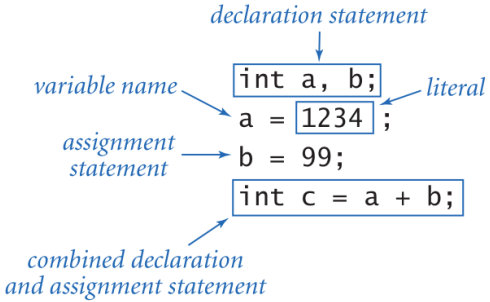
Variables, Identifiers, Literals, Declaration, and Assignment

- **Variables**
  - A **named memory location** in which data can be **stored or retrieved**
  - Data must match with the data type specified in the variable declaration
- **Identifier**
  - Identifiers are **names** of a variable, method, class, object, package or interface etc.
- **Literals**
  - A literal is the **source code representation of a fixed value**
  - for example, `boolean result = true;` //here `true` is a literal for boolean data type



Variables, Identifiers, Literals, Declaration, and Assignment *(Cont'd)*

- **Declaration**
  - A declaration statement is used to **declare** a variable by specifying its **data type and name**
  - A declaration allocates memory cell for a variable
- **Assignment**
  - **Assigns** or stores a value to a variable
- **Initialization**
  - Assigning **initial value** of a variable



2 Objects and Classes

Objects and Classes

- **Classes**
  - **Classes** are **templates** or **blueprints** for **Objects**
  - **Data** and **methods** are defined within Classes
- **Objects**
  - An **object** is an **Instance** of a class
  - The process of **creating** an object is called **instantiation**
  - The **attributes** of an object are called **instance variables**
  - The **methods** of **an object** are called **instance methods**
  - Objects are created using **`new`** keyword

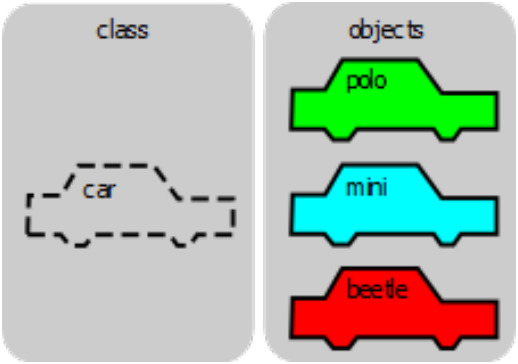


Figure 1: Class and Objects. Figure Courtesy - Wikimedia Commons

Objects and Classes *(Cont'd)*

- An example simple `Car` class

```
1  public class Car {
2          //properties
3          private String modelName;
4          private String modelColor;
5          private int modelMaxSpeed;
6          private int currentSpeed;
7  }
```

- Creating a `Car` object
- Note that, we are using **`new`** keyword and calling default constructor method, which is **`Car()`**

```
1  Car c1 = new Car(); //creating a car by default constructor
```

Objects and Classes *(Cont'd)*

- Let's write a **constructor** method of `Car` class
- A constructor is a **special method** that is called when an object is instantiated

```
1  public class Car {
2          //properties
3          private String modelName;
4          private String modelColor;
5          private int modelMaxSpeed;
6          private int currentSpeed;
7
8  /*
9  *Constructor
10 */
11 public Car(String name, String color, int mSpeed, int speed){
12         this.modelName = name;
13         this.modelColor = color;
14         this.modelMaxSpeed = mSpeed;
15         this.currentSpeed = speed;
16 }
17
18 }
```

Objects and Classes *(Cont'd)*

- Let's create another car by calling our constructor

```
Car c2 = new Car("Mini Coopers", "Cyan", 140, 0);
```

- Let's add a method called `printCarInfo()`

```
public void printCarInfo(){
System.out.println("\n-: Car Information :-");
System.out.println("Model: " + this.modelName);
System.out.println("Color: " + this.modelColor);
System.out.println("Max Speed: " + this.modelMaxSpeed + " km/h");
System.out.println("Current Speed: " + this.currentSpeed + " km/h");
}
```

- Let's write code to call `printCarInfo()` method from `main()` to see the car info

```
c2.printCarInfo();
```

Objects and Classes *(Cont'd)*

Listing 1: Car.java

```
1  public class Car {
2  //properties
3  private String modelName;
4  private String modelColor;
5  private int modelMaxSpeed;
6  private int currentSpeed;
7
8  public Car(){
9  //default constructor
10 }
11
12 /* Constructor */
13 public Car(String name, String color, int mSpeed, int speed){
14 this.modelName = name;
15 this.modelColor = color;
16 this.modelMaxSpeed = mSpeed;
17 this.currentSpeed = speed;
18 }
19
20 public void printCarInfo(){
21 System.out.println("\n-: Car Information :-");
```

Objects and Classes *(Cont'd)*

```
22  System.out.println("Model: " + this.modelName);
23  System.out.println("Color: " + this.modelColor);
24  System.out.println("Max Speed: " + this.modelMaxSpeed + " km/h");
25  System.out.println("Current Speed: " + this.currentSpeed + " km/h");
26  }
27  }
```

- Let's implement two more methods -

```
1  void accelerate(int speed);
2  void brake(int speed);
```

Thanks

Thanks for your time and attention!

kamruddin@aiub.edu