2.A little dive into the data types

Sakib Abrar

CSE

Bangladesh University of Engineering & Technology

sakib.cghs@gmail.com

September 6, 2020

Overview

Data types

Data type simply indicates data that will be stored in our programs. But not all the data is same. That's why we need to clearly specify to java which type of data we are going to store.

We store our data in variables (also known as identifiers) and there are several types of it.

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in the memory (RAM, SWAP ect.).

Variables that've been used in our program will reside in the memory as long as the program is running.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Major Classification

- **Primitive Data Type :** Primitive data types are more primal data types that doesn't need any object oriendted approach to be stored. Java developers included these data types to maintain the portability of java as the size of these primitive data types do not change from one operating system to another.

- **Non Primitive Data Type :** Non-Primitive type or Reference types are created using defined constructors of the classes. They are used to access objects. We'll see them when we start studying Object Oriented Features.

Primitive Data types

- Integers
  – byte 8-bit integer (new)
  – short 16-bit integer
  – int 32-bit signed integer
  – long 64-bit signed integer
- Real Numbers
  – float 32-bit floating-point number
  – double 64-bit floating-point number
- Other types
  – char 16-bit, Unicode 2.1 character
  – boolean true or false, false is not 0 in Java

Declaration and assignment

```
public class DataTypeExamples {
    public static void main(String[] args) {
        byte x;  // a byte sized variable
        x = 120;

        int num;
        num = 10; // only interger numbers allowed

        float pi; // floating point numbers allowed

        double PI; // also floating point number but bigger
        PI = 3.1415965;

        char ch; // a single UNICODE character
        ch = 'a';

        boolean passed; // either true or false
        passed = false;
    }
}
```

Don’t forget to comment your code

It is very important to comment your code

Comments are ignored by the compiler so they don’t get run when executed.
A good piece of code should be explained using comments.

Two types of commenting in java

- Single line comment: starts with // and ends in only one line
- Multi line comment: enclosed with /* and */ and between these two you can write as many line as you want

Printing Data types

Printing any primitive data type in java is very easy.
Just put it inside **System.out.println()**

```
public class DataTypeExamples {
    public static void main(String[] args) {
        byte x;  // a byte sized variable
        x = 120;
        System.out.println(x);
        int num = 10; // interger numbers allowed
        System.out.println(num);
        double PI; // also floating point number but bigger
        PI = 3.1415965;
        System.out.println(PI);
        char ch = 'a';
        System.out.println(ch);
        boolean passed;
        passed = false;
        System.out.println(passed);
    }
}
```

Time for user input

First you need to import Scanner. (WTH is importing?)

```
import java.util.Scanner;

public class DataTypeExamples {
    public static void main(String[] args) {

        Scanner ourScanner = new Scanner(System.in);
        /* spoiler alert!! this is a non-primitive type.
        All the non-primitive type variable should
        usually be newed before using*/

        int num; // for integer input .nextInt();
        num = ourScanner.nextInt();
        double pi; // for double input .nextDouble();
        pi = ourScanner.nextDouble();

        System.out.println("Inputted␣integer␣:" + num);
        System.out.println("Inputted␣double␣:␣" + pi);
    }
}
```

Importing a class

Don't try to code everything of your own.

Be smart and only think about coding your own portion! Leave the rest to someone else and learn to use it.

JDK includes lots of java classes that may fulfill your needs.

Besides there are plenty of free and opensource commonly solved code of java online.

Just import and use.

Type casting

Type casting is when you assign a value of one primitive data type to another type.
In Java, there are two types of casting:

- Widening Casting (automatically) - converting a smaller type to a larger type size
  byte → short → char → int → long → float → double
- Narrowing Casting (manually) - converting a larger type to a smaller size type
  double → float → long → int → char → short → byte

Type casting variations

Type casting means converting one type of variable into another type of variable.

- **Widening Casting :**
  Widening casting is done automatically when passing a smaller size type to a larger size type.

- **Narrowing Casting :**
  Narrowing casting must be done manually by placing the type in parentheses in front of the value.

Type casting Example

```java
public class MyClass {
  public static void main(String[] args) {

    int myInt = 9;
    double myDouble = myInt;
    // Automatic casting: int to double
    System.out.println(myInt);        // Outputs 9
    System.out.println(myDouble);     // Outputs 9.0

    double myDouble = 9.78;
    int myInt = (int) myDouble;
    // Manual casting: double to int
    System.out.println(myDouble);     // Outputs 9.78
    System.out.println(myInt);        // Outputs 9
  }
}
```

The End