SOT
SCHOOL OF TECHNOLOGY

OBJECT ORIENTED PROGRAMMING WITH JAVA (20CP204T)

Presented by:

Dr. Shivangi K. Surati

Assistant Professor,
Department of Computer Science and Engineering,
School of Technology,
Pandit Deendayal Energy University

Outline

- Data Types
- Type Conversions
- Type Promotion in Expressions
- Scope of Variables
- Main Method
- Print Method
- Scan Variables
- Size and Type of Variables

Data Types

- Java is strongly typed
  - Every variable and expression has a type
  - Every type is strictly defined
  - Type compatibility check in each assignment (direct or through parameter passing in method calls)
- Primitive data types
  - Eight primitive data types (byte, short, int, long, char, float, double and boolean)
- All data types have a strictly defined range, regardless of particular platform

Data Types…

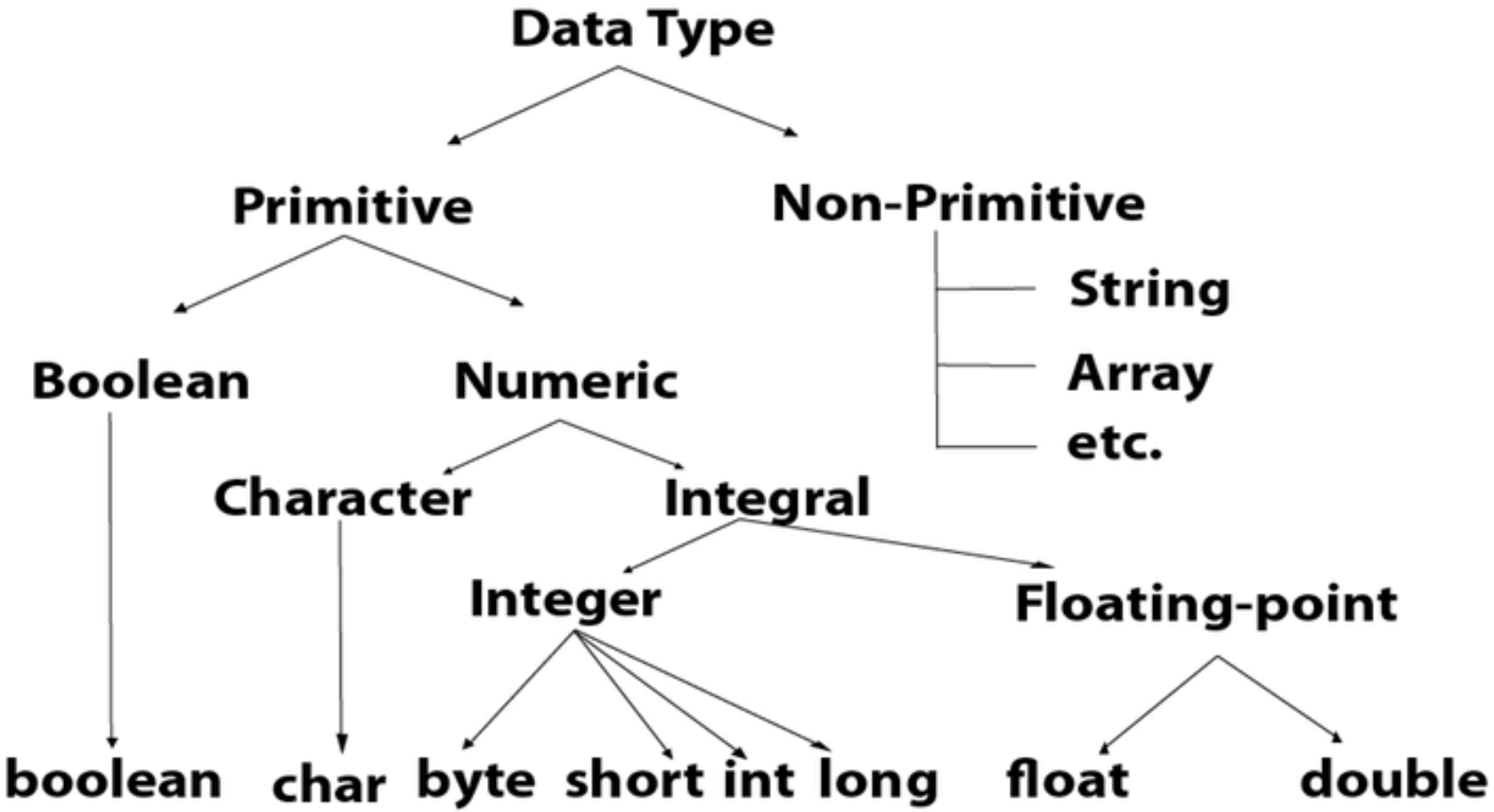

Image source: Java Data Types - Javatpoint

Data Types- Integers

- byte, short, int, long
- all are signed, positive and negative
- Java does not support unsigned integers

Type	Contains	Data Types	Size	Range
byte	Signed integer	Byte	8 bit or 1 byte	-128 to 127
short	Signed integer	Short	16 bit or 2 bytes	-32,768 to 32767
int	Signed integer	Int	32 bit or 4 bytes	-2147,483,648 to 2147,483,647
long	Signed integer	Long	64 bit or 8 bytes	-263 to 263-1 or -9223,372,036,854,755,808 to 9223,372,036,854,755,807

8:18

Data Types…

- Floating Point types
  - float, double
  - Stores numbers with fractional precision
- Characters – char, supports ASCII conversion, no negative char
- Booleans – Boolean – takes values true or false

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false

Data Types and Variables- Examples

```
(1) short s, t;
    s=55;
(2) int lightspeed,
    long seconds;
    lightspeed = 20500;
    seconds = 1000*24*60*60;
(3) char ch1, ch2;
    ch1=65;
    ch2='B';
    System.out.print("ch1 and ch2: " + ch1 + " " +ch2);
(4) boolean b=false;
    if (b) System.out.println("in true block");
```

O/P: ?

```
System.out.println("10 > 5 is "+ (10 > 5));
```

Find Output

```
public class Main {
   public static void main(String[] args) {
          int num1 = 10;
          int num2 = 20;
          long num3 = 0;
          num3 = num1 + num2 * 10 + Character.SIZE;
          System.out.println(num3);
   }
}
```

Type Conversions

- Automatic (implicit type conversion)
  - The two types are compatible
  - The destination type is larger than the source type

- Casting incompatible types (explicit type conversion)
  - (target-type) value
  - int a = 257; byte b; double d = 323.142;

    b = (byte) a;

    a = (int) d;

    b = (byte) d;

Type Promotion in Expressions

- In expression, the range of intermediate values will sometimes exceed the range of either operand

Ex: byte a=40, b=50, c=100;

int d = a*b + c;

- Thus, Java automatically promotes each byte, short or char operand to int during expression evaluation
  - So a*b = 40*50 = 2000 is data type int, not byte
  - Also data type of result of a*b + c is int
- Compile-time error !

Ex: byte b=50;

byte a = b*2; //Error, can't assign int to byte

- Solution?

byte a = (byte) b*2;

Scope of Variables

- Variables can be declared in any block – before its use

```
cnt =0;
int cnt;
```
} Error !

- Block begins with an opening curly bracket
- Scope of variable – within the block in which it is declared

Find Output

```
public class Main {
   public static void main(String[] args) {
          int i;
          for(i=1; i<= 3; i++)
          {
                 int x = -1;
                 System.out.println(“x is ”+x);
                 x = 10;
                 System.out.println(“now x is ”+x);
          }
   }
}
```

Find Output…

```
public class Institutes {
    public static void main(String[] args) {
        String name = “PDEU";
        {
            System.out.println(name + " Institute");
            name = “pdeu";
        }
        System.out.println(name + “ Institute");
    }
}
```

Find Output…

```
public class VarScope {
    public static void main(String[] args) {
        int x = 10;
        {
            int y = 20;
            System.out.print(x + ", " + y);
        }
        {
            y = 10;
            x = 15;
            System.out.print(" - " + x + ", " + y);
        }
        System.out.print(" - " + x + ", " + y);
    }
}
```

Find Output…

```
public class ScopeOfVariables {
    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        {
            System.out.print(x + ", " + y);
        }
        {
            x = 15;
            System.out.print(" - " + x + ", " + y);
        }
        System.out.print(" - " + x + ", " + y);
    }
}
```

Main Method

public static void main (String[] args)

- public: This is the access modifier of the main method. It has to be public so that java runtime can execute this method.
- static: When java runtime starts, there is no object of the class present. That's why the main method has to be static so that JVM can load the class into memory and call the main method (there is only one static variable per class).
- void: Java programming mandates that every method provide the return type. Java main method doesn't return anything, that's why it's return type is void.
- main: This is the name of java main method. It's fixed and when we start a java program, it looks for the main method.

Print Method

- used to display a text on the console

1. public void print(String s)
   - an overloaded method of the PrintStream class
   - Call: System.out.print(parameter);

     Can't create object of PrintSteam class directly, so use instance of the PrintStream class that is System.out

2. public void println(String s)
   - an overloaded method of the PrintStream class

3. public PrintStream printf(String format, Object... args)
   - an overloaded method of the PrintStream class
   - print the formatted string to the console using the specified format string and arguments

Scan Variables

- Scanner method:

```
import java.util.Scanner;
class Program{
        public static void main(String []args){
                Scanner myObj = new Scanner(System.in);
                String userName = myObj.nextLine();
                int i = myObj.nextInt();
        }
}
```

Size and Type of Variables

- Get size of variable- size operator
  - EX: size of integer : Integer.SIZE/8

    size of character: Character.SIZE/8

- Get type of variable
  - Call getClass().getSimpleName() via the object
  - EX: String str;

```
System.out.println(str.getClass().getSimpleName());
int i;
System.out.println(((Object)i).getClass().getSimpleName());
```