Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 B, NAB

Consultation: Appointment by e-mail

BITS Pilani

Pilani Campus

Programming Syntax

BITS Pilani
Pilani Campus

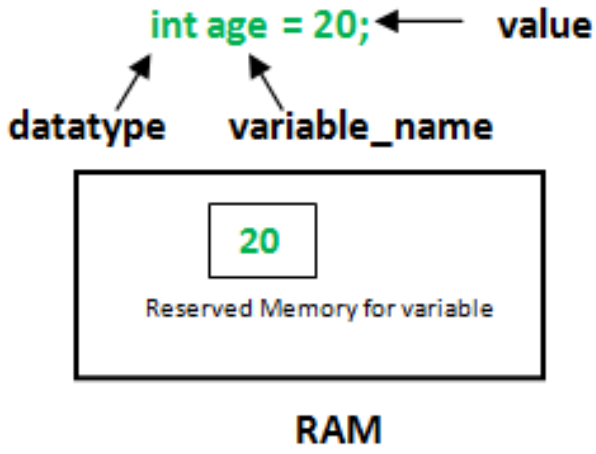
Naming Convention (~~Rule~~)

- Used to name your identifiers such as class, package, variable, constant, method etc.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Variable

- Name given to a memory location. It is the basic unit of storage in a program.
  - The value can be changed during program execution.
  - All the operations done on the variable effects that memory location.
  - Variables must be declared before they can be used. (but not necessarily in the beginning)
- Types of variables
  - Static or class variables
  - Instance Variables
  - Local variables



Static Variables

- A *class variable* is any field declared with the static modifier

- Tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated.

Static Variables-Example

```
public class StaticVarExample {
   static String myClassVar="class or static variable";

   public static void main(String args[]){
      StaticVarExample obj1 = new StaticVarExample();
      StaticVarExample obj2 = new StaticVarExample();

      System.out.println(obj1.myClassVar);
      System.out.println(obj2.myClassVar);

      //changing the value of static variable using obj2
      obj2.myClassVar = "Changed Text";

      System.out.println(obj1.myClassVar);
      System.out.println(obj2.myClassVar);  } }
```

System.out.println(myClassVar);

Output:
class or static variable
class or static variable
Changed Text
Changed Text

Static variable across classes- Example


```
class Zero{
static String classvar = "Static Variable of another class";
}
class First{
    public static void main(String args[]){
     System.out.println(Zero.classvar);
    }
}
```

Where static variables can be used?

- Counting the number of objects
- Specify the college or dept name for all the student objects
- Specify the branch name and code for all account holders of a particular branch.

Instance (Non-static) Variables

- Objects store their individual states in "non-static fields"
- Fields declared without the static keyword.
- Non-static fields are also known as *instance variables* because their values are unique to each *instance* of a class.

Instance Variables-Example

```
public class InstanceVarExample {
   String myInstanceVar="instance variable";

   public static void main(String args[]){
      InstanceVarExample obj1 = new InstanceVarExample();
      InstanceVarExample obj2 = new InstanceVarExample();

      //Two objects will display "class or static variable"
      System.out.println(obj1.myInstanceVar);
      System.out.println(obj2.myInstanceVar);

      //changing the value of static variable using obj2
      obj2.myInstanceVar = "Changed Text";

      //All will display "Changed Text"
      System.out.println(obj1.myInstanceVar);
      System.out.println(obj2.myInstanceVar);  } }
```

Output:
instance variable
instance variable
instance variable
Changed Text

Local Variable

- Defined within a block or method or constructor
- These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.
- The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.

Local Variable-Example

```
public class VariableExample {

// instance variable
public String myVar="instance variable";

public void myMethod(){
// local variable
String myVar = "Inside Method";
System.out.println(myVar); }

public static void main(String args[]){

// Creating object
VariableExample obj = new VariableExample();
System.out.println("Calling Method");

obj.myMethod();
System.out.println(obj.myVar);
}
 }
```

Output:
Calling Method
Inside Method
Instance variable

Primitive Data Types

innovate achieve lead

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

Note: There is no `sizeof` operator in Java as there is in C. Size of each primitive is fixed

Type Conversion

- When the value of one data type is assigned to another, the two types might not be compatible with each other.
- If the data types are compatible, then Java will perform the conversion automatically known as Automatic Type Conversion
- If not then they need to be casted or converted explicitly. For example, assigning an int value to a long variable.

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

Double –> Float –> Long –> Int –> Short –> Byte

Narrowing or Explicit Conversion

Type Conversion - Example

```
char ch = 'c';
int num = 88;
num = ch;   // Automatic type conversion
System.out.println(num);
```

Ans: 99

```
ch = num;
System.out.println(ch);
```

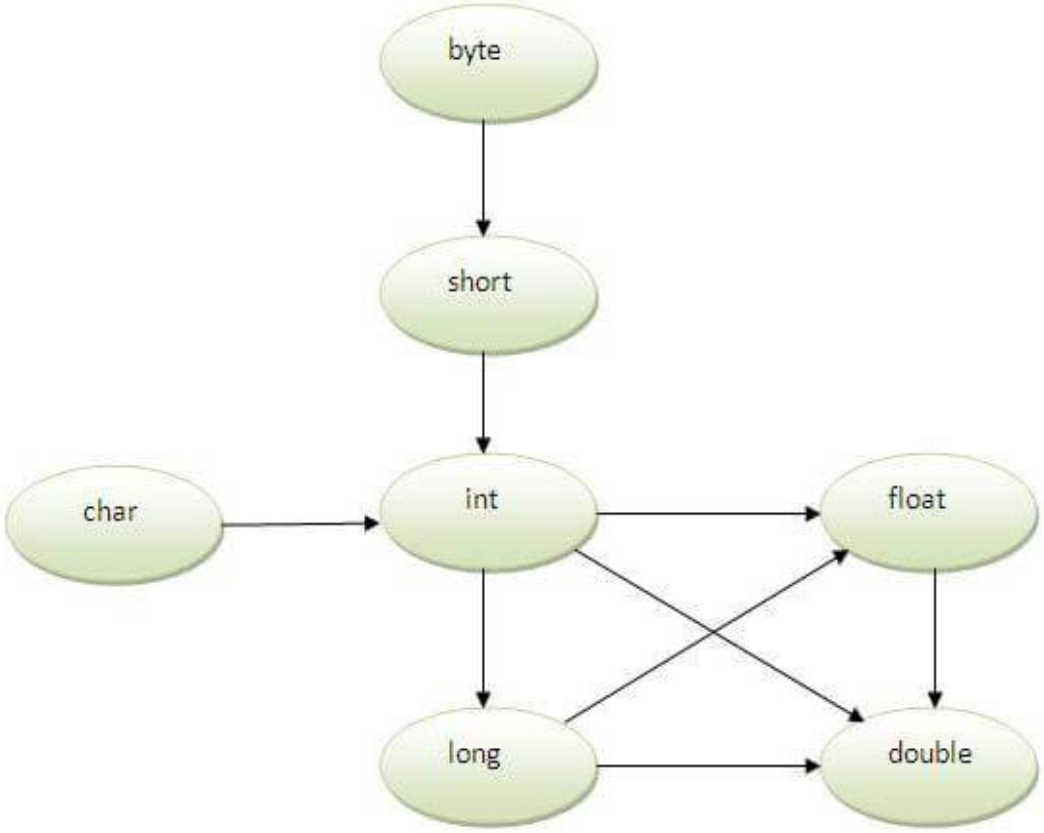
Error: incompatible types: possible lossy conversion from int to char

Solution: ch = (char) num; //Type casting or explicit conv.

Ans: X

Type Promotion

- While evaluating expressions, the intermediate value may exceed the range of operands and hence the expression value will be promoted.
- Some conditions for type promotion are:
  - Java automatically promotes each byte, short, or char operand to int when evaluating an expression.
  - If one operand is a long, float or double the whole expression is promoted to long, float or double respectively.



Type Promotion-Example


```
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = .1234;

// The Expression
double result = (f * b) + (i / c) - (d * s);
System.out.println("result = " + result);
```

Additional Example


```
byte b = 50;
b = (b * 2);
System.out.println(b);
```

Error: due to type promotion

Solution: b = (byte) (b * 2); //Explicit conversion

Ans: 100

Primitive Data Types -String

- It is not technically a primitive data type
- Java programming language provides special support for character strings via the java.lang.String class.
- Enclosing your character string within double quotes will automatically create a new String object;
  - **String s = "this is a string";**

Operators, Precedence, Associativity

innovate achieve lead

Operators	Associativity	Type
++ --	right to left	unary postfix
++ -- + - ! (*type*)	right to left	unary prefix
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&	left to right	boolean logical AND
^	left to right	boolean logical exclusive OR
|	left to right	boolean logical inclusive OR
&&	left to right	conditional AND
||	left to right	conditional OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Example-Associativity

```
int a=10,b=20,c=30,d=40,e=50;
int result = a>=10 ? b <20 ?c :d :e ;
System.out.println("Result:" + result );
```

- ***Recall: Associativity of Ternary Operator is R to L***
- ***Answer: 40***

Example-Conditional AND

int a=-9;

Boolean b=**true;**

Boolean result = (a>0) && (b=**false) ;**

System.***out.println("Result:" + result +"b="+b);***

Recall: Conditional AND is a short circuit operator

Answer: b= true