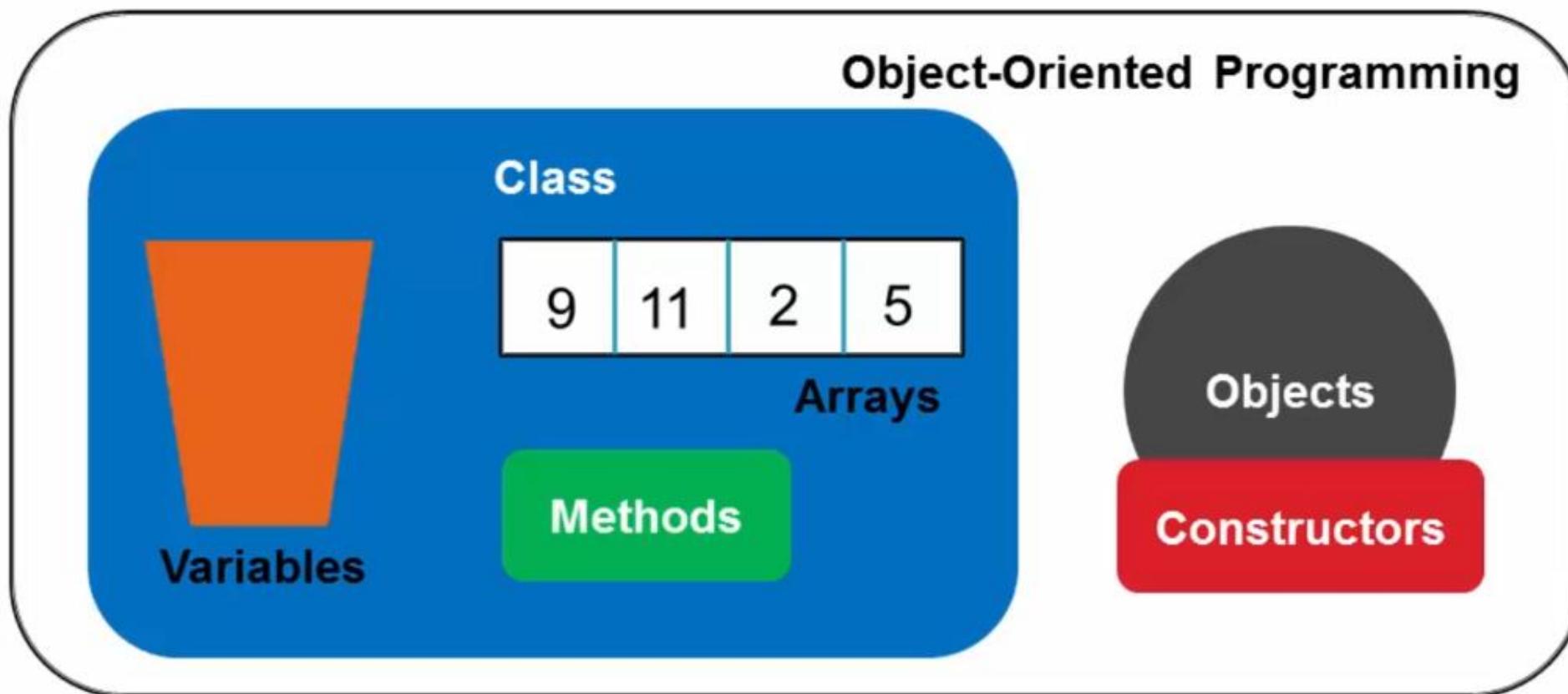


Classes, Objects and their Members

Dheeru Mundluru



Agenda



Absolute Basics

Let us understand :

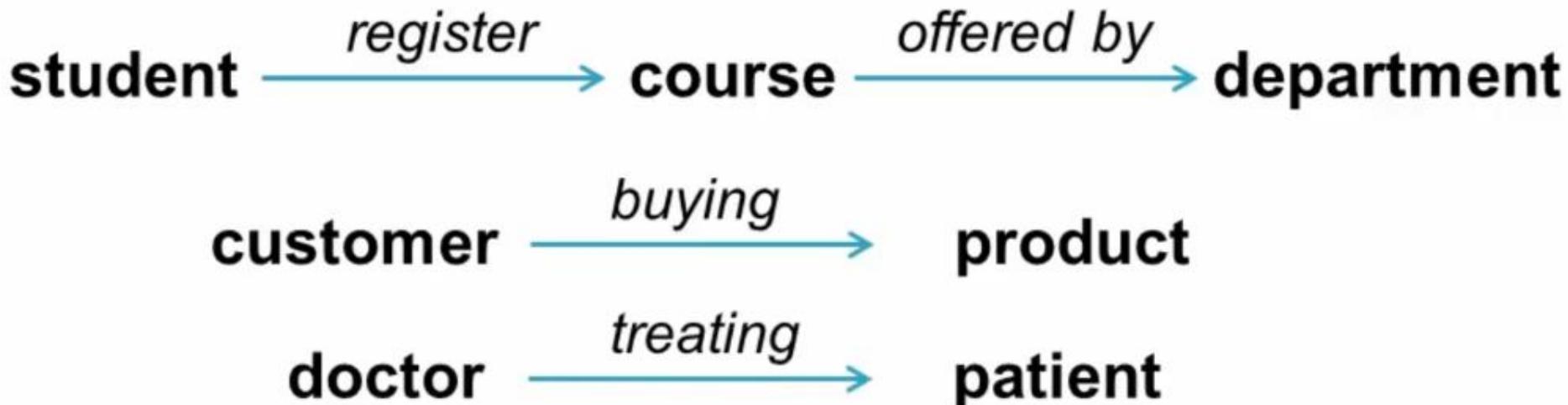
1. Class
2. Object
3. Variables
4. Array
5. Methods
6. Constructor

Class & Objects

Dheeru Mundluru

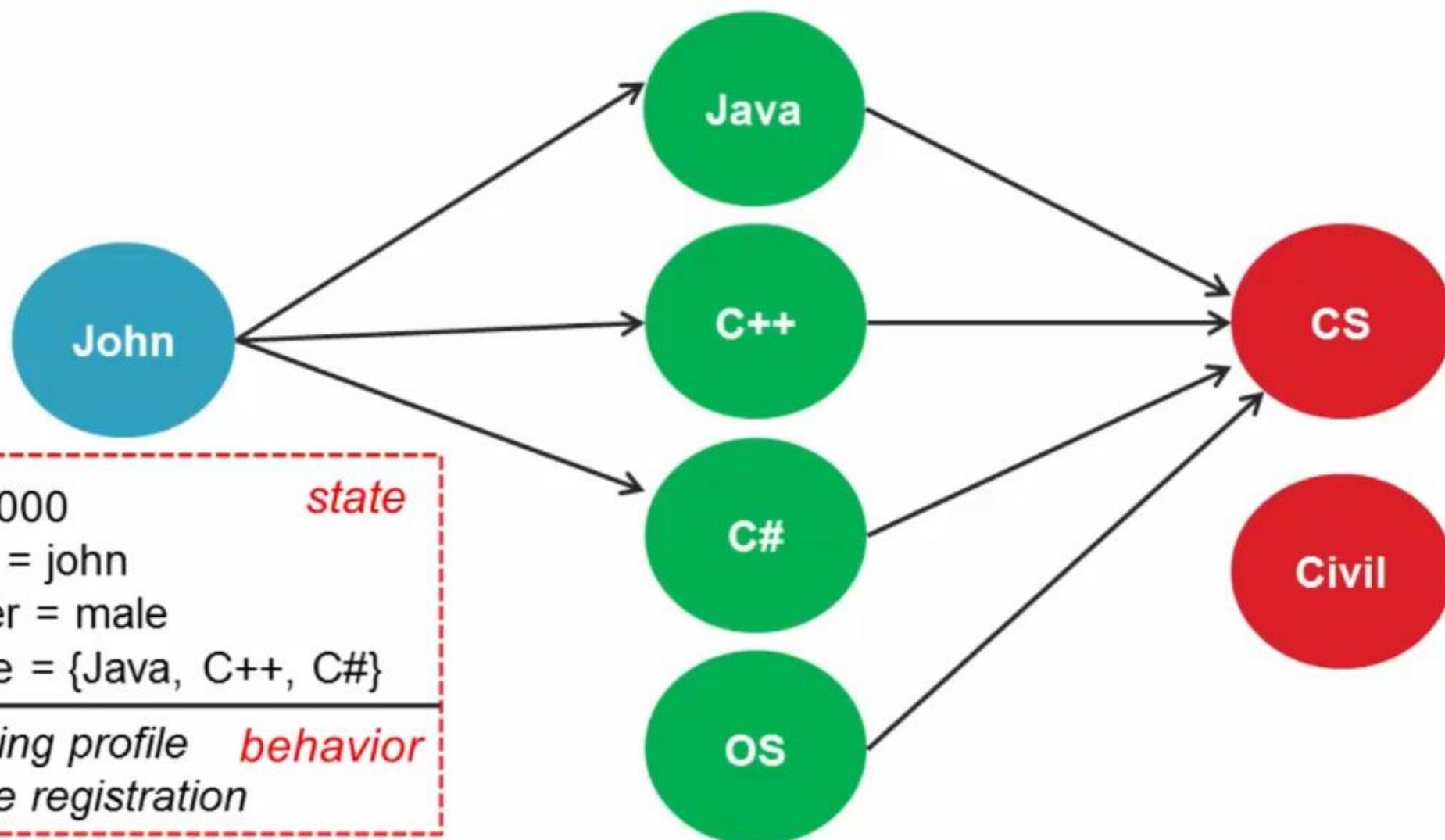
Object-Oriented Programming

- ▶ Roots in 1960s
- ▶ Implement *large projects* in *simple way*
- ▶ Model ***real-world scenarios*** in a more ***natural*** way



An object oriented programming language (OOPs) helps model real world scenarios in a more natural way with the help of objects and classes.

student $\xrightarrow{\text{register}}$ **course** $\xrightarrow{\text{offered by}}$ **department**



This is a realistic scenario where , the **student John**, has registered in three **courses (Java , C++ , C#)** which are offered by the **department CS**.

Let's understand John in details..

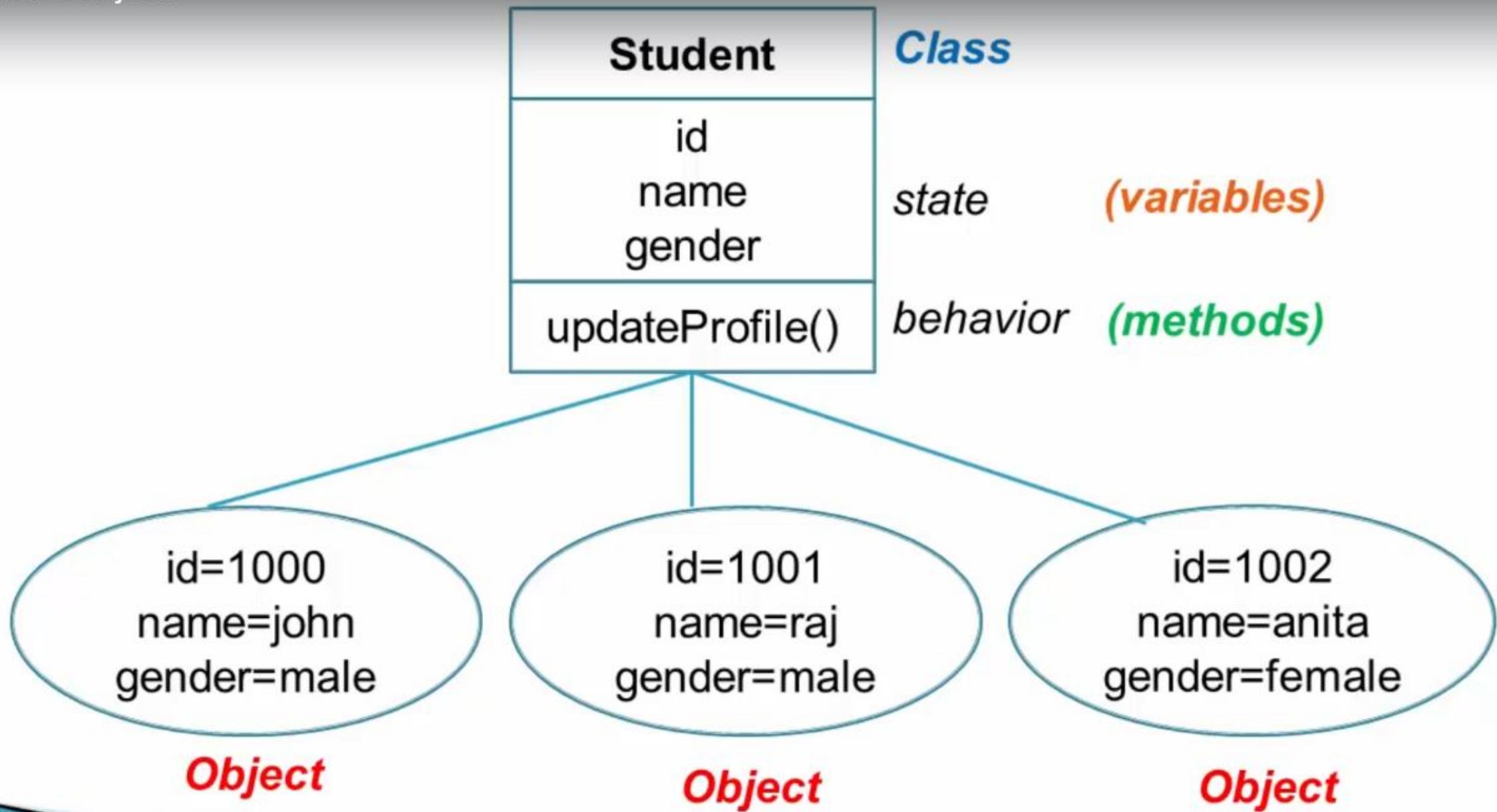
John has some data/state : *id , name , gender* etc.

In OOPs , Data in John's instance is referred as ***the State of the instance***.

John can also perform the two actions *register a course* and *update his profile*

In OOPs, The actions are referred as ***behavior/Method of the instance***.

The whole instance of *John* is referred to as an ***Object***.



Similarly to John object, we can have other object with similar set of state and behavior as mentioned in Student class.

Class is like a blueprint.

Objects/instance are actual implementation of the class.

```
class Student {  
    // variable declarations  
    int id;  
    String name;  
    String gender;  
  
    // method definitions  
    boolean updateProfile(String newName) {  
        name = newName;  
        return true;  
    }  
}
```

Here is the **definition of Student class**

- **Object state** is defined by **variables**
- **Object behavior** is defined by the **method** (updateProfile)
- Variables and methods in a class are considered as **members of the class**.
- Each variable has **Data type**, to segregate type of data a variable hold or can hold.

```
class StudentTest {  
    public static void main (String[] args) {  
        // 1. creating a new student object  
        Student s = new Student();  
  
        // 2. setting student's state  
        s.id = 1000;  
        s.name = "joan";  
        s.gender = "male";  
  
        // 3. updating profile with correct name  
        s.updateProfile("john");  
    }  
}
```

```
boolean updateProfile(String newName) {  
    name = newName; ←  
    return true;  
}
```

Let's us see, how to create Object of a class, for this let us create another class with main method this time.

Object creation generally involved the keyword **new**.

Example : Student s = new Student();

Part “**new Student();**” : Creates an object of the Student class (by calling it’s default constructor).

Part **Student s** : Assign the created object to s(variable name), whose datatype is Student class.
Combination of these parts, results an object s of Student class

To set object’s state (set the values for the variables) for that, we use the dot operator.

Example : s.id , s.name etc.

To invoke or call object’s method , we will use dot operator again.

Example : s.updateProfile(“John”)



java new 2 new 3

```
class 1BasicsDemo {  
    // 1. Naming rules for classes/methods/variables  
        // a) First character: letter, underscore, $  
        // Remaining: letter, underscore, $, numbers  
        // b) No reserved keywords (See resources section)  
  
    // 2. **Java is case-sensitive  
  
    // 3. Printing to console  
  
    // 4. Comments or Disabling Code  
  
    // 5. Arithmetic Operations  
}
```

let's look at some absolute basics before proceeding further.

Naming rules for classes, methods and variables :

- first character of the name has to be either a letter or underscore or dollar. If you have something else, then we will get a compilation error
- The subsequent characters will be letter , underscore , dollar and numbers also.
- No reserved keywords can be used.

Note : Java has around 50 reserved keywords, like class , new etc



java new 2 new 3

```
class 1BasicsDemo {  
    // 1. Naming rules for classes/methods  
        // a) First character:  
        //     Remaining: letters, numbers  
        // b) No reserved keywords  
  
    // 2. **Java is case-sensitive  
  
    // 3. Printing to console  
  
    // 4. Comments or Disabling Code  
  
    // 5. Arithmetic Operations  
  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:2: error: <identifier> expected  
    class 1BasicsDemo {  
           ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 1 :

Let's put a number in front of the class.

Let's compile it

Here it is, it says that this is not right.



java new 2 new 3

```
class BasicsDemo {  
    // 1. Naming rules for classes/methods  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int lid = 0;  
  
    // 2. **Java is case-sensitive  
  
    // 3. Printing to console  
  
    // 4. Comments or Disabling Code  
  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:2: error: <identifier> expected  
    class 1BasicsDemo {  
           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
           ^  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
           ^  
2 errors  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 2 :

Similarly, create some variables say

Int 1id = 0;

Again compiled error.



java new 2 new 3

```
class BasicsDemo {  
    // 1. Naming rules for classes/  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int _id = 0;  
  
    // 2. **Java is case-sensitive  
  
    // 3. Printing to console  
  
    // 4. Comments or Disabling Code  
  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:2: error: <identifier> expected  
    class 1BasicsDemo {  
           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
          ^  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
          ^  
2 errors  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>_
```

Example 3 :

Let's put an _ (underscore) in variable's start

It compiled fine.



Demo.java new 2 new 3

```
class BasicsDemo {  
    // 1. Naming rules for classes/methods:  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int id1 = 0;  
  
    int class = 0;  
  
    // 2. **Java is case-sensitive  
    // 3. Printing to console  
    // 4. Comments or Disabling Code  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:2: error: <identifier> expected  
    class 1BasicsDemo {  
           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
          ^  
BasicsDemo.java:7: error: <identifier> expected  
    int iid = 0;  
          ^  
2 errors  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:9: error: <identifier> expected  
    int class = 0;  
          ^  
BasicsDemo.java:9: error: <identifier> expected  
    int class = 0;  
          ^  
BasicsDemo.java:19: error: reached end of file while parsing  
    ^  
3 errors  
  
C:\javaindepth\src\com\semanticsquare\basics>
```

We cannot use any reserved keywords

Example 4 :

Let's put 'class' as variable name.

This will give Compilation error will be : we cannot use this as a reserve keyword,



Demo.java new_2 new_3

```
class BasicsDemo {  
    // 1. Naming rules for classes/methods:  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int id1 = 0;  
  
    int id1 = 0;  
  
    // 2. **Java is case-sensitive  
    // 3. Printing to console  
    // 4. Comments or Disabling Code  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:9: error: variable id1 is already defined in class BasicsDemo  
        int id1 = 0;  
                ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>
```

We cannot have duplicate names.

Example 5 :

For instance, we cannot create another variable called id1



Demo.java new 2 new 3

```
class BasicsDemo {  
    // 1. Naming rules for classes/methods  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int id1 = 0;  
  
    void foo() {}  
    void foo() {}  
  
    // 2. **Java is case-sensitive  
    // 3. Printing to console  
    // 4. Comments or Disabling Code  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:9: error: variable id1 is already defined in class BasicsDemo  
        int id1 = 0;  
                ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:10: error: method foo() is already defined in class BasicsDemo  
        void foo() {}  
                  ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>
```

Similarly, **we cannot also have duplicate methods** with same types of arguments.

Example 6 :

Let create two

```
void foo() {}
```

Compilation error : “method foo() already defined in the class.

23. Demo: Absolute Java Basics

```
class BasicsDemo {  
    // 1. Naming rules for classes/methods  
        // a) First character:  
        //     Remaining: letters  
        // b) No reserved keywords  
    int id1 = 0;  
  
    // 2. **Java is case-sensitive  
  
    int id = 0;  
    int Id = 0;  
    int ID = 0;  
  
    void foo() {}  
    void Foo() {}  
  
    // 3. Printing to console  
  
    // 4. Comments or Disabling Code  
  
    // 5. Arithmetic Operations  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:9: error: variable id1 is already defined in class BasicsDemo  
        int id1 = 0;  
                ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:10: error: method foo() is already defined in class BasicsDemo  
        void foo() {}  
                    ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>_
```

Java is case sensitive

Example 1 :

Let's define variables id, Id and ID.

This would work fine.

Example 2 :

Similarly, you can also have methods.

`void foo() {}`

`void Foo() {}`

This would work fine too.



Demo.java new 2 new 3

```
int ID = 0;

void foo() {}
void Foo() {}

// 3. Printing to console
// Adapted from http://www.ntu.edu.sg/~m330001/Java/JavaBasics.htm
static void print() {
    System.out.println("\n\nInside print...");
    System.out.println("Hello, world!!!");
    System.out.println();
    System.out.print("Hello, world!!!");
    System.out.println("Hello, world!!!");
    System.out.print("Hello, world!!!");
    System.out.print("Hello, world!!!");
}

public static void main(String[] args) {
    print();
}

// 4. Comments or Disabling Code
// 5. Arithmetic Operations
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside print...
Hello, world!!!
Hello, world!!!Hello,
world!!!
C:\javaindepth\src\com\semanticsquare\basics>
```

© semanticsquare.com

Next, **print something to the console.**

Print method under System language just print the argument passed to it.

Println method , print and then shift the cursor to new line.

Demo: Absolute Java Basics

```
int ID = 0;

void foo() {}
void Foo() {}

// 3. Printing to console
// Adapted from http://www.ntu.edu.sg/home/ehchua/programming/java/J1a_Introduction.htm
static void print() {
    System.out.println("\n\nInside print ...");
    System.out.println("Hello, world!!"); // Advance cursor to beginning of next line
    System.out.println(); // Print empty line
    /*System.out.print("Hello, world!!"); // Cursor stayed after the printed string
    System.out.println("Hello,");
    System.out.print(" ");
    System.out.print("world!!");*/
}

public static void main(String[] args) {
    print();
}

// 4. Comments or Disabling Code

// 5. Arithmetic Operations
```

Next one is the comments or disabling code from compilation:

- Single line comments :

double slash (forward slash) means ignore everything after that on that line.

- Multi line (or block) comments :

It starts with a slash and asterisk : /* and ends with reverse of it (*/)

The whole block inside it will be disabled.

Variables: Introduction

Dheeru Mundluru

data type

int id = 1000;

primitive variable

String name = "john";

object reference

Student s = new Student();

object reference

Statically typed language → Static type checking

Dynamically typed language → Dynamic type checking

Variables, are containers that hold some data.

Every variable has a **Data type (Variable type)** associated with it, data type decides what type of data a variable can hold.

If the data type is int, it can hold numeric data (like 1000)

If the type is string, it can hold textual data (like John)

If the type is a Class, then the variable can hold a object like student object in this case.

In Java, a **data type cannot be changed** once variable is declared. Hence Java is called **statically typed language** (as type is fixed/static).

Static type checking allows earlier detection of data type related issues.

In Dynamically typed language, you will find the issue at runtime itself and at that point it is too late.

Primitive variable :

The variable holding a raw data like 1000, (example : id = 1000) can be considered as primitive variable.

Object reference :

The variable which point/refer an object is called object reference.

Example 1:

```
Student s = new Student();
```

Example 2:

```
String name = "john"; (As String is a class in Java the associated variable is considered as object reference )
```

data type

`int id = 1000;`

primitive variable

`String name = "john";`

`interface Student s = new Student();` **object reference**

Statically typed language → Static type checking

Dynamically typed language → Dynamic type checking

21 people have written a note here.

It would also be an object reference if the variable type is something called an interface.

Variable Declaration

- ▶ **<type> <name> [= *literal or expression*];**
 - ▶ literal → raw data
 - int id = 1000;
 - boolean flag = true;
 - String name = "john";
 - ▶ expression → evaluated to single value
 - int id = x;
 - int id = x + y;
 - Student s = new Student();
 - ▶ Can appear **anywhere** in class
- 
- declaration statements**

Literal is simply raw data.

Example :

- Numeric value 1000 to integer literal
- Boolean literal true
- String literal John

Expression is something that gets evaluated to a single value.

- x is an expression
- x + y
- student object creation part new Student();

Reinitializing Variable

- ▶ Variable ~ *something whose value can be changed*
- ▶ **Assignment statement**
 - <name> = literal or expression;
 - count = 23;
 - count = x + y;
 - *cannot appear at class-level*

The variable declared at **class level** and **has no static keyword** associated with it , then they are called as **Instance variable**.

The variable declared at **class level** and **has static keyword** associated with it , then they are called as **Static variable**.

The variable declared at **non class level**, then they are called as **Local variable**.

Reinitialization, it is similar to a declaration statement, but without the variable type as the type has already been defined.

Example :

count = 23;

count = x+y;

- Reinitialization statements **cannot appear at class level** directly. They can appear inside members of the class, such as methods.

Default value to the variable can only work **at class level** directly, that means ; Initialization is optional only at class level.

The screenshot shows a Java development environment with a code editor and a terminal window.

Code Editor:

```
class Student {  
    int id;  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println(id);  
        System.out.println(nextId)  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Command Prompt:

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
0  
1
```

Page Footer:

© semanticsquare.com

Udemy

Let's see the demo of reinitialization and default value of variables:

We will create a new class called Student.

Example 1 :

Id at class level , without any value assignment, should get default value of integer (that is 0)

The screenshot shows a Java development environment with a code editor and a terminal window.

Code Editor:

```
class Student {  
    void compute() {  
        int id;  
        int nextId = id + 1;  
        System.out.println(id);  
        System.out.println(nextId)  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Terminal Window:

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
0  
1  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:6: error: variable id might not have been initialized  
            int nextId = id + 1;  
                    ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 2 :

Id at method level (non-class level or method variable or local variable) , without any value assignment will not any get default value.

It gives a compilation error. "variable id not initialized"

```
file demo.java Student.java Student.java BasicsDemo.java Student.java Command Prompt C:\javaindepth\src\com\semanticsquare\basics>javac Student.java C:\javaindepth\src\com\semanticsquare\basics>java Student 0 1 C:\javaindepth\src\com\semanticsquare\basics>javac Student.java Student.java:6: error: variable id might not have been initialized int nextId = id + 1; 1 error C:\javaindepth\src\com\semanticsquare\basics>javac Student.java C:\javaindepth\src\com\semanticsquare\basics>java Student 1000 1001 C:\javaindepth\src\com\semanticsquare\basics>javac Student.java C:\javaindepth\src\com\semanticsquare\basics>java Student 1000 1001 C:\javaindepth\src\com\semanticsquare\basics>javac Student.java Student.java:3: error: <identifier> expected id = 1000; 1 error C:\javaindepth\src\com\semanticsquare\basics>
```

The screenshot shows a Java code editor and a terminal window. The code editor displays a Java class named 'Student' with a main method. The variable 'id' is declared but not initialized. The terminal window shows the compilation and execution of this code. It first compiles the code successfully, then runs it, outputting '0' and '1'. When recompiled, it shows an error for the undeclared variable 'id'. Finally, it runs the code again, outputting '1000' and '1001', demonstrating that the variable was initialized to 1000 by the previous run.

Example 3 :

Let's try reinitialization or assignment statement at class level using id = 1000;

It gives compilation error too.

25. Demo: Declaring & Re-initializing Variables

```
class Student {  
    int id = "Dheeru";  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println(id);  
        System.out.println(nextId)  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:6: error: incompatible types: String cannot be converted to int
 int id = "Dheeru";
 ^
1 error

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:2: error: incompatible types: String cannot be converted to int
 int id = "Dheeru";
 ^
1 error

C:\javaindepth\src\com\semanticsquare\basics>

Example 4:

Initialization of variable with different data type (using id = "Dheeru") gives compilation error(incompatible types), because Java is a statically type language.

25. Demo: Declaring & Re-initializing Variables

```
class Student {  
    int id = 1000;  
  
    void compute() {  
        id = "Dheeru"; // Error: incompatible types: String cannot be converted to int  
        int nextId = id + 1;  
  
        System.out.println(id);  
        System.out.println(nextId)  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:6: error: incompatible types: String cannot be converted to int  
        id = "Dheeru";  
               ^  
1 error
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 5:

Similarly Reinitialization of variable with different data type (using id = "Dheeru") also gives compilation error(incompatible types).

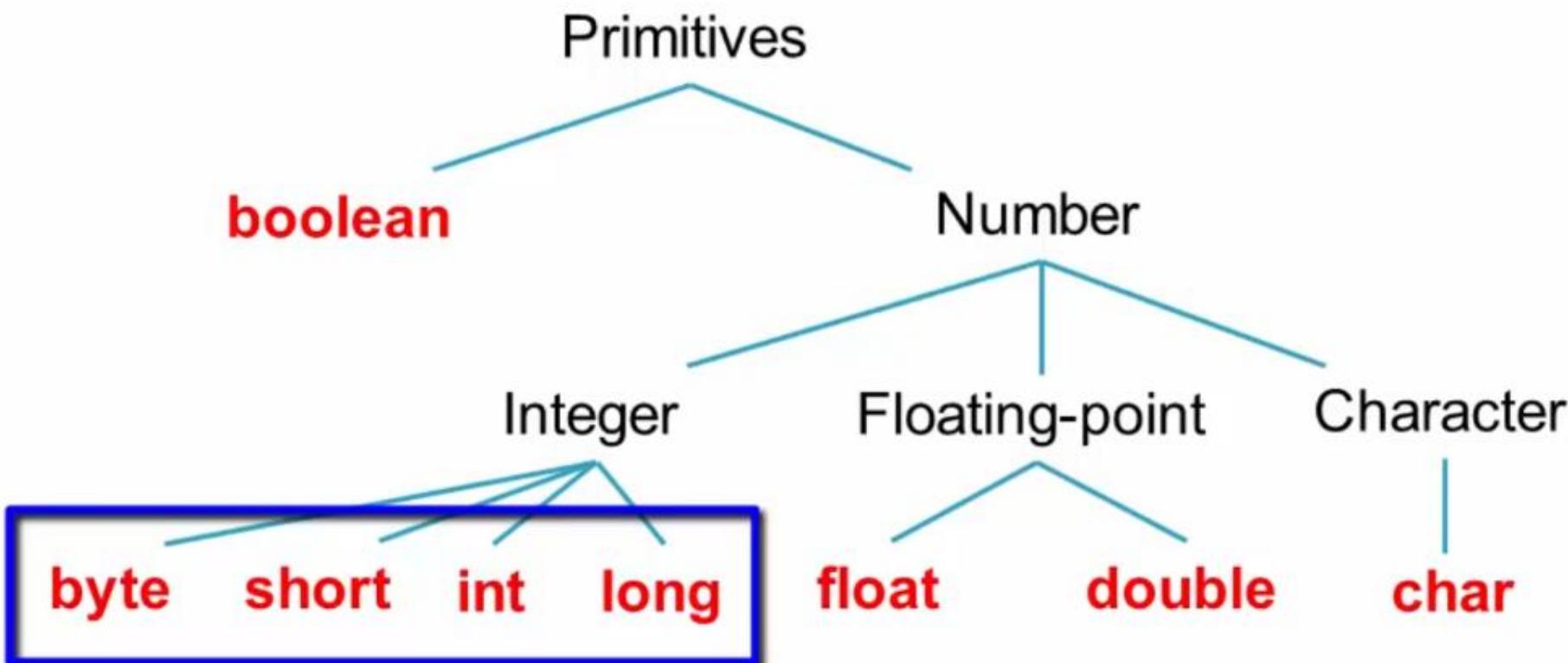
Variables: Primitive Types

Dheeru Mundluru

Unauthorized copying, distribution or display of this course material is
punishable under law

Primitive Types

8 primitives types



Java is object oriented programming language , however for performance reasons, Java has 8 primitive data types , to fit different set of data and utilize memory efficiently.

- Boolean
- Byte, Short , Int , Long
- Float, Double
- Char
- Except for Boolean, the rest of the data types are numeric.
- Even char, which can be used to represent an alphabet, is a numeric type (it is internally represented as unsigned integer).

Integer Data Types

Dheeru Mundluru

Integers

- ▶ Whole or fixed-point numbers, e.g., 65, -1000
- ▶ *byte, short, int, long*

Type	Bit depth	Value range	Default
byte	8 bits	-2^7 to $2^7 - 1$	0
short	16 bits	-2^{15} to $2^{15} - 1$	0
int	32 bits	-2^{31} to $2^{31} - 1$	0
long	64 bits	-2^{63} to $2^{63} - 1$	0

Integers (positive/negative whole numbers) can be stored in 4 primitive data types:

Byte

Short

Int

Long

This table shows their :

1. bit depth (number of memory bits to store), base is 2 is used for range because computers use binary system which deals with only 0 and 1.
2. The number range which can be stored.
3. All four Integer data types have 0 as a default value.
4. The left most bit is called as **signed bit or most significant bit**, If sign bit = 1 , it means negative number.

So let's look at what we have here for each of the data types.

Byte Range = -128 to +127

Short Range = -32768 to 32767

Int Range = - 2,147,483,648 to 2,147,483,647

Long Range = -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Note : Even though we have 4 data types. There are only 2 type of literal

1. **Integer Literal** = No need to add anything in the value. This literal is compatible with Int , Short and Byte data type
2. **Long Literal** = Need to add L in the value. This literal is compatible with Long data type only.

```
JarHello.java IODemo.java Student.java Student.java

class Student {
    int id = 1000;

    byte age = 18;

    void compute() {
        int nextId = id + 1;

        System.out.println("id: " + id);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.compute();
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

Example 1 :

```
byte age = 18;
```

We already used int data type , let's try Byte. It works fine.

27. Primitive Variables: Integers + Demo

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    byte rank = 165; //[-128, 127]  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Now let's introduce another variable, called rank of data type byte.

So let's assume that there is some entrance test to get an admission in the university and so there would be a rank associated with the entrance test.

So that said, the student has got the rank of 165.

```
JarHello.java IODemo.java Student.java Student.java

class Student {
    int id = 1000;

    byte age = 18;
    byte rank = 165; //[-128, 127]

    void compute() {
        int nextId = id + 1;

        System.out.println("id: " + id);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.compute();
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:5: error: incompatible types: possible lossy conversion from int to
      byte
          byte rank = 165;
                           ^
1 error
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 2 :

```
byte rank = 165;
```

The value is out of range for byte so it should give “value is too large” error ,
However it gives us an error, “incompatible types, possible lossy conversion” because the
default literal is Integer

27. Primitive Variables: Integers + Demo

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    short rank = 165; // [-128,  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

Example 3 :

short rank = 165;

So let's just change the rank to short type, because shot has a larger range. It would work.

```
JarHello.java IODemo.java Student.java Student.java

class Student {
    int id = 1000;

    byte age = 18;
    short rank = 165; //[-128,
    int phone = 2234567890;

    void compute() {
        int nextId = id + 1;

        System.out.println("id:");
        System.out.println("nextId:");
        System.out.println("age:");
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.compute();
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:6: error: integer number too large
        int phone = 2234567890;           ^
                                                1 error
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 4 :

```
int phone = 2234567890;
```

It gives error “number too large”, so this number falls outside the range of int. since data type and literal type is same it will not give “incompatible type” error

27. Primitive Variables: Integers + Demo

```
class Student {  
    int id = 1000;  
  
    byte age = 1_8;  
    short rank = 165; // [-128,  
    long phone = 223_456_7890L;  
  
    // Integer Literals: int Li  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id:  
        System.out.println("nex  
        System.out.println("age  
        System.out.println("pho  
    }  
  
    public static void main(Str  
        Student s = new Student  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
C:\javaindepth\src\com\semanticsquare\basics>
```

we can have underscore to make value more readable, underscore can only be added in between the numbers ,it cannot appear at the beginning or end.

Example 5 :

```
long phone = 223_456_7890L
```

We need to make it a long literal by adding **small l** or **capital L** and change the data type to long literal.

```
JarHello.java IODemo.java Student.java Student.java

class Student {
    int id = 1000L;

    byte age = 18L;  I
    short rank = 165; //[-128,
    int phone = 2234567890L;

    void compute() {
        int nextId = id + 1;

        System.out.println("id: " + nextId);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.compute();
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:2: error: incompatible types: possible lossy conversion from long to int
        int id = 1000L;
                           ^
Student.java:4: error: incompatible types: possible lossy conversion from long to byte
        byte age = 18L;
                           ^
Student.java:6: error: incompatible types: possible lossy conversion from long to int
        int phone = 2234567890L;
                           ^
3 errors
```

Example 6 :

```
int id = 1000L;  
byte age = 18L;  
int phone = 2234567890L;
```

Since long literal can not assign to Int, Short , Byte data types , we will get other type of error,
“incompatible types” even if the assigned value is within range.

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    short rank = 165; //[-128, 127]  
    long phone = 223_456_7890L; // Java 7 -- readability  
  
    // Integer Literals: int literal, Long Literal  
  
    int minValue = Integer.MIN_VALUE;  
    int maxValue = Integer.MAX_VALUE;  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Boxed Primitive


```
JarHello.java IODemo.java Student.java Student.java BasicsDemo.java Student.java Student.java Student.java my.ini Student.java BasicsDemo.java

class Student {
    int id = 1000;

    byte age = 18;
    short rank = 165; //[-128, 127]
    long phone = 223_456_7890L; // Java 7 -- readability

    // Integer Literals: int literal, Long Literal

    int minValue = Integer.MIN_VALUE;
    int maxValue = Integer.MAX_VALUE;

    void compute() {
        int nextId = id + 1;

        System.out.println("id: " + id);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
        System.out.println("phone: " + phone);
        System.out.println("minValue: " + minValue);
        System.out.println("maxValue: " + maxValue);
    }

    public static void main(String[] args) {
        Student s = new Student();
```

Example 7 :

```
int minValue = Integer.MIN_Value;  
int maxValue = Integer.MAX_Value;
```

To see the maximum and minimum value of any primitive data type, we can use it's **boxed primitive class** or **wrapper class** comes with the Java Library.

```
JarHello.java  IODemo.java  Student.java  Student.java

class Student {
    int id = 1000;

    byte age = 18;
    short rank = 165; //[-128,
    long phone = 223_456_7890L;

    // Integer Literals: int Li
    int minValue = Integer.MIN_
    int maxValue = Integer.MAX_

    void compute() {
        int nextId = id + 1;

        System.out.println("id: " + id);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
        System.out.println("phone: " + phone);
        System.out.println("minValue: " + minValue);
        System.out.println("maxValue: " + maxValue)
    }

    public static void main(String[] args) {
        Student s = new Student();
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java

C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
phone: 2234567890

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:4: error: cannot find symbol
          byte age = _1_8;
                           ^
      symbol:   variable _1_8
      location: class Student
1 error

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java

C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
phone: 2234567890
minValue: -2147483648
maxValue: 2147483647
```



```
class BasicsDemo {  
    // Adapted from http://www.ntu.edu.sg/home/ehchua/programming/java/J1a_Introduction.html  
    static void print() {  
        System.out.println("\n\nInside print ...");  
        System.out.println("Hello, world!!"); // Advance cursor to beginning of next line  
        System.out.println(); // Print empty line  
        System.out.print("Hello, world!!"); // Cursor stayed after the printed string  
        System.out.println("Hello,");  
        System.out.print(" "); // Print a space  
        System.out.print("world!!");  
    }  
  
    static void primitives() {  
        System.out.println("\n\nInside primitives ...");  
        long intHex = 0x0041L; // 16 power 0 * 1 + 16 power 1 * 4  
        System.out.println("intHex: " + intHex);  
  
        // Java 7  
        int intBinary = 0b0100_0001;  
        System.out.println("intBinary: " + intBinary);  
  
        int intOctal = 0101;  
        System.out.println("intBinary: " + intBinary);  
    }  
}
```



Integer and long literal are not limited to just Decimal number system (base 10), they can be use other number systems like Hexadecimal (0x) , Binary (0b) and Octal (0)

Example :

```
long intHex = 0x0041L;  
int intBinary = 0b0100_0001;  
int intOctal = 0101;
```

```
JarHello.java  IODemo.java  Student.java  Student.java

class BasicsDemo {
    // Adapted from http://www.
    static void print() {
        System.out.println("\n");
        System.out.println("Hello");
        System.out.println();
        System.out.print("Hello");
        System.out.println("Hello");
        System.out.print(" ");
        System.out.print("world");
    }

    static void primitives() {
        System.out.println("\n");
        long intHex = 0x0041L;
        System.out.println("intHex: " + intHex);

        // Java 7
        int intBinary = 0b0100_1001;
        System.out.println("intBinary: " + intBinary);

        int intOctal = 0101;
        System.out.println("intOctal: " + intOctal);
    }
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside primitives ...
intHex: 65
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside primitives ...
intHex: 65
intBinary: 65
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside primitives ...
intHex: 65
intBinary: 65
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside primitives ...
intHex: 65
intBinary: 65
intOctal: 65
```


Floating-point Data Types

Dheeru Mundluru

Floating-point Numbers

- ▶ Real numbers, e.g., 3.14, -0.014
- ▶ *float, double*

Type	Bit depth	Value range	Default	Precision
float	32 bits	-3.4E38 to 3.4E38	0.0f	6-7 decimal digits
double	64 bits	-1.7E308 to 1.7E308	0.0d	15-16 decimal digits

Floating point numbers

- Real numbers like 3.14 and they can be negative, too.
- Floating point can be represented by either **float** data type (with 6-7 digit precision) or **double** data type (with 15-16 digit precision).
- float or double has default of zero.
- Just like Int literal is default for Integer number, **Double literal** is **default literal** for floating point numbers
- the floating literals can be represented by f at the end of the number.
- the double literals can be represented by d at the end of the number optionally. By default floating point number will be considered as double literals only.

>Edit Search View Encoding Language Settings Macro Run

student.java Student.java Student.java myini Student.java

```
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     float gpa = 3.8f;  
8  
9     void compute() {  
10         int nextId = id + 1;  
11  
12         System.out.println("id: " + id);  
13         System.out.println("nextId: " + nextId);  
14         System.out.println("age: " + age);  
15         System.out.println("phone: " + phone);  
16         System.out.println("gpa: " + gpa);  
17     }  
18  
19     public static void main(String[] args) {  
20         Student s = new Student();  
21         s.compute();  
22     }  
23 }
```

source file

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 1:

Float gpa = 3.8f;

It compile fine and prints 3.8

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     float gpa = 3.8;  
8  
9     void compute() {  
10         int nextId = id + 1;  
11  
12         System.out.println("id: " + id);  
13         System.out.println("nextId: " + nextId);  
14         System.out.println("age: " + age);  
15         System.out.println("phone: " + phone);  
16         System.out.println("gpa: " + gpa);  
17     }  
18  
19     public static void main()  
20         Student s = new Student();  
21         s.compute();  
22     }  
23 }  
source file
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:7: error: incompatible types: possible lossy conversion from double  
          to float  
          float gpa = 3.8;  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 2:

```
float gpa = 3.8;
```

Since the default floating point literal is double.

It gives compilation error “Incompatible Types, Possible lossy conversion from double to float”.

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     double gpa = 3.8;  
8  
9     void compute() {  
10        int nextId = id + 1;  
11  
12        System.out.println("id: " + id);  
13        System.out.println("nextId: " + nextId);  
14        System.out.println("age: " + age);  
15        System.out.println("phone: " + phone);  
16        System.out.println("gpa: " + gpa);  
17    }  
18  
19    public static void main()  
20        Student s = new Student();  
21        s.compute();  
22    }  
23 }  
source file
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:7: error: incompatible types: possible lossy conversion from double  
          to float  
          float gpa = 3.8;  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 3:

```
double gpa = 3.8;
```

It compile fine and prints 3.8

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     double gpa = 3.8d;  
8  
9     void compute() {  
10        int nextId = id + 1;  
11  
12        System.out.println("id: " + id);  
13        System.out.println("nextId: " + nextId);  
14        System.out.println("age: " + age);  
15        System.out.println("phone: " + phone);  
16        System.out.println("gpa: " + gpa);  
17    }  
18  
19    public static void main(String[] args) {  
20        Student s = new Student();  
21        s.compute();  
22    }  
23 }  
source file
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:7: error: incompatible types: possible lossy conversion from double  
          to float  
          float gpa = 3.8;  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 4:

```
double gpa = 3.8d;
```

It compile fine and prints 3.8 too.

Edit Search View Encoding Language Settings Macro Run

student.java Student.java Student.java myini Student.java

```
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     double gpa = 3.8e4;  
8  
9     void compute() {  
10         int nextId = id + 1;  
11  
12         System.out.println("id: " + id);  
13         System.out.println("nextId: " + nextId);  
14         System.out.println("age: " + age);  
15         System.out.println("phone: " + phone);  
16         System.out.println("gpa: " + gpa);  
17     }  
18  
19     public static void main(  
20         String[] args)  
21     {  
22         Student s = new Student();  
23         s.compute();  
24     }  
25 }
```

source file

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java

C:\javaindepth\src\com\semanticsquare\basics>java Student

id: 1000
nextId: 1001
age: 18
phone: 2234567890
gpa: 38000.0

C:\javaindepth\src\com\semanticsquare\basics>

We can also use scientific notation or exponential notation.

Example 5:

```
double gpa = 3.8e4;
```

It compile fine and prints 38000.0

Edit Search View Encoding Language Settings Macro Run Plugins Window



source file

length : 553 lines : 2

Ln:7 Col:64 Sel:0

Dos\Windows

ANSI

let's look at the precision

Example 6:

```
double gpa = 3.88888888888888888888888889999999999999;
```

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     double gpa = 3.888888888888889;  
8  
9     void compute() {  
10         int nextId = id + 1;  
11  
12         System.out.println("id: " + id);  
13         System.out.println("nextId: " + nextId);  
14         System.out.println("age: " + age);  
15         System.out.println("phone: " + phone);  
16         System.out.println("gpa: " + gpa);  
17     }  
18  
19     public static void main(String[] args) {  
20         Student s = new Student();  
21         s.compute();  
22     }  
23 }
```

source file

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 38000.0
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.888888888888889
```

rounding

```
C:\javaindepth\src\com\semanticsquare\basics>
```

it will be truncated to 15 digits because double has a precision of 15, along with rounding off the last digit to 9.

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1 class Student {  
2     int id = 1000;  
3  
4     byte age = 18;  
5     long phone = 223_456_7890;  
6  
7     float gpa = 3.888888888888889;  
8  
9     void compute() {  
10         int nextId = id + 1;  
11  
12         System.out.println("id: " + id);  
13         System.out.println("nextId: " + nextId);  
14         System.out.println("age: " + age);  
15         System.out.println("phone: " + phone);  
16         System.out.println("gpa: " + gpa);  
17     }  
18  
19     public static void main(String[] args) {  
20         Student s = new Student();  
21         s.compute();  
22     }  
23 }
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 38000.0  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.888888888888889  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.888888888888888  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 7:

```
float gpa = 3.888888888888888888888888999999999999f;
```

it will be truncated to 6 digits because float has a procession of 6-7 digits.

Floating-point Pitfalls

Dheeru Mundluru

Edit Search View Encoding Language Settings Macro Run Plugins Window ?

30. Demo: Floating-point Pitfalls

Student.java Student.java myini Student.java BasicsDemo.java new_4

```
1     System.out.print("world!!");  
2 }  
3  
4 static void primitives() {  
5     System.out.println("\n\nInside primitives ...");  
6     int intHex = 0x0041; // 16 power 0 * 1 + 16 power 1 * 4  
7     System.out.println("intHex: " + intHex);  
8  
9     // Java 7  
0     int intBinary = 0b0100_0001;  
1     System.out.println("intBinary: " + intBinary);  
2  
3     int intOctal = 0101;  
4     System.out.println("intOctal: " + intOctal);  
5 }  
6  
7 public static void main(String[] args) {  
8     // Language Basics 1  
9     //print();  
0     //primitives();  
1     System.out.println(1 - 0.9);  
2  
3 }
```

length : 1184 lines : 34 Ln : 31 Col : 39 Sel : 0 Dos\Windows ANSI CC IN

1x 1:54 / 10:55 © semanticsquare.com

If your application needs exact results for floating point numbers , then you should avoid float or double data types.

Computers uses format (IEEE 754 binary floating point format), which cannot represent floating point accurately.

It's an issue with all programming languages.

The issue is because the denominator of these numbers are not a power of two.

For instance,

0.1 , it is $1/10$. Ten in the denominator is not a power of 2.

0.2 , it is $1/5$ and 5 is not a power of 2.

However,

0.5 , it is $1/2$, so it can be accurately represented now, but most of the numbers cannot be.

Example 1:

```
System.out.println(1 - 0.9);
```

Edit Search View Encoding Language Settings Macro Run



Student.java Student.java Student.java myini Student.java

```
1     System.out.print("w
2 }
3
4     static void primitives()
5         System.out.println(
6             int intHex = 0x0041
7             System.out.println(
8
9             // Java 7
10            int intBinary = 0b00000000000000000000000000000001
11            System.out.println(
12
13            int intOctal = 0101
14            System.out.println(
15
16
17     public static void main()
18         // Language Basics
19         //print();
20         //primitives();
21         System.out.println(
22
23 }
```

source file

"Concourse"

Command Prompt

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
0.09999999999999998

C:\javaindepth\src\com\semanticsquare\basics>

It prints this value at 0.0999999998, instead of correct value 0.1



student.java Student.java Student.java myini Student.java BasicsDemo.java new 4

```
1     System.out.print("world!!");  
2 }  
  
3 static void primitives() {  
4     System.out.println("\n\nInside primitives ...");  
5     int intHex = 0x0041; // 16 power 0 * 1 + 16 power 1 * 4  
6     System.out.println("intHex: " + intHex);  
7  
8     // Java 7  
9     int intBinary = 0b0100_0001;  
10    System.out.println("intBinary: " + intBinary);  
11  
12    int intOctal = 0101;  
13    System.out.println("intOctal: " + intOctal);  
14 }  
  
15 public static void main(String[] args) {  
16     // Language Basics 1  
17     //print();  
18     //primitives();  
19     System.out.println(1 - 0.9);  
20     System.out.println(0.1 + 0.2)  
21 }  
22 }
```

Example 2:

```
System.out.println(0.1 + 0.2);
```

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
1     System.out.print("w  
2 }  
3  
4 static void primitives()  
5     System.out.println(  
6         int intHex = 0x0041  
7     System.out.println(  
8  
9 // Java 7  
0     int intBinary = 0b00000000000000000000000000000001  
1     System.out.println(  
2  
3     int intOctal = 0101  
4     System.out.println(  
5 }  
6  
7 public static void main(  
8     // Language Basics  
9     //print();  
0     //primitives();  
1     System.out.println(  
2     System.out.println(  
3 }  
source file
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.09999999999999998  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.09999999999999998  
0.30000000000000004  
C:\javaindepth\src\com\semanticsquare\basics>
```

It prints this value at 0.30000000000004, instead of correct value 0.3



If we are dealing with precision/money , don't use float

Example:

we want to apply some discount percentage

```
double prize = 1000;
```

For new price mathematically both the below approaches give result as 100, but computer does not.

- System.out.println(price - (price * 0.9))
- System.out.println(price * (1 - 0.9))

```
8 // Java 7
9 int intBinary = 0b101010;
10 System.out.println(intBinary);
11
12 int intOctal = 0101010;
13 System.out.println(intOctal);
14
15 }
16
17 public static void main(String[] args) {
18     // Language Basics
19     //print();
20     //primitives();
21     System.out.println("Hello World!");
22     System.out.println(0.1);
23     // 0.1 ~ 0.00011001
24
25     double price = 1000;
26     double discountPercent = 10;
27     double discountAmount = price * discountPercent / 100;
28     System.out.println(discountAmount);
29     System.out.println(price - discountAmount);
30 }
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
0.0999999999999998

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
0.0999999999999998
0.3000000000000004

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
0.0999999999999998
0.3000000000000004
100.0

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
0.0999999999999998
0.3000000000000004
100.0
99.9999999999997
```

Computer gives 99.9999999997 for the below approach

- `System.out.println(price * (1 - 0.9))`



It is recommended to use **BigDecimal** class for floating points.

Down side is , BigDecimal is slow.

Here we create object of floating point numbers before using them.

Object uses methods (like .add) to perform arithematic operations

Example :

```
BigDecimal first = new BigDecimal("0.1");
BigDecimal second = new BigDecimal("0.2");
```

```
System.out.println( first.add(second) );
```

```
Edit Search View Encoding Language Settings Macro Run  
Student.java Student.java Student.java myini Student.java  
4     System.out.println()  
5 }  
6  
7 public static void main()  
8     // Language Basics  
9     //print();  
10    //primitives();  
11    System.out.println()  
12    System.out.println()  
13    // 0.1 ~ 0.00011001  
14  
15    double price = 1000.0  
16    double discountPercent = 10.0  
17    double discountAmount = price * discountPercent / 100.0  
18    System.out.println(price)  
19    System.out.println(discountAmount)  
20  
21    BigDecimal first = new BigDecimal("0.1")  
22    BigDecimal second = new BigDecimal("0.00011001")  
23    System.out.println(first.add(second))  
24 }  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.0999999999999998  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.0999999999999998  
0.3000000000000004  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.0999999999999998  
0.3000000000000004  
100.0  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.0999999999999998  
0.3000000000000004  
100.0  
99.9999999999997  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
0.0999999999999998  
0.3000000000000004  
100.0  
99.9999999999997  
0.3
```


Character Data Type

Dheeru Mundluru

Characters

- ▶ Single letter characters, e.g., 'A', '0', '\$'
- ▶ *char*
 - char degree = 'B'; (66) [0,65535]
- ▶ Data representation ~ **16-bit unsigned integer**

Type	Bit depth	Value range	Default
char	16 bits	0 to $2^{16} - 1$	'\u0000'

char data type to hold **single letter characters**(not multiple characters) like an alphabet or a digit like zero or a special symbol like \$.

The bit depth is 16 bits unsigned integers(only positive numbers internally), so the range will 0 to 65535.

The default value of char is **null** character.

Characters

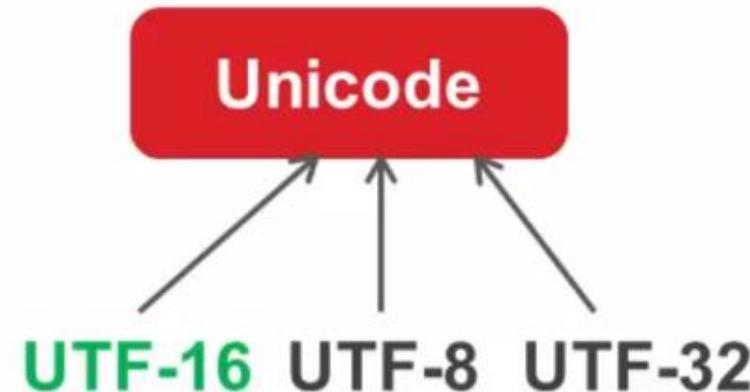


- ▶ Single letter characters, e.g., 'A', '0', '\$'

- ▶ *char encoding scheme*

- char degree = 'B'; (66) '\u0042'

- ▶ Data representation ~ **16-bit unsigned integer**



Type	Bit depth	Value range	Default
char	16 bits	0 to $2^{16} - 1$	'\u0000'

- ▶ 'B' → 0042 → 00000000 01000010 (66)

<'B', \u0042, 66>

Java uses UTF-16 encoding scheme (like C# and Python).

Example ,

the character **B** would be encoded as UTF 16 hexadecimal number : **0042** and stored as 0000
0000 0100 0010 binary format.



```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = 'B';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Example 1 :

char degree = 'B'

```
testDrive.java TapeDeck.java JamHall.java ClientA.java ClientB.java  
  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = 'B';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

It prints B.

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = B;  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: cannot find symbol
```

```
    char degree = B;                                ^  
          symbol:   variable B  
          location: class Student  
1 error
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

Let's remove quotes

Example 2 :

char degree = B

Now, the compiler will think B is a variable, so compilation error will be “can not find symbol”.

```
testDrive.java TapeDeck.java JamHall.java ClientA.java ClientB.java  
  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = 'BA';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: cannot find symbol  
          char degree = B;  
                  ^  
         symbol:   variable B  
         location: class Student  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: unclosed character literal  
          char degree = 'BA';  
                           ^  
Student.java:9: error: unclosed character literal  
          char degree = 'BA';  
                           ^  
2 errors  
  
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 3 :

```
char degree = 'BA'
```

Compilation error will be “**Unclosed character literal**” .

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = '';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: cannot find symbol  
          char degree = B;
```

```
      symbol: variable B  
      location: class Student  
1 error
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: unclosed character literal  
          char degree = 'BA';
```

```
Student.java:9: error: unclosed character literal  
          char degree = 'BA'__;
```

```
2 errors
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: empty character literal  
          char degree = '';
```

```
1 error
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

It cannot be empty too

Example 4 :

```
char degree = "
```

Compilation error will be "**Empty character literal**" .

```
testDrive.java TapeDeck.java JarFile.java ClientA.java ClientB.java  
  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = ' ';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree:  
C:\javaindepth\src\com\semanticsquare\basics>
```

Although you can have an empty space here.

Example 5 :

```
char degree = ''
```

Compilation fines and print a space .

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = '\u0042';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree:
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

We can also use escape sequence (like '\u0042')

Example 6 :

```
char degree = '\u0042'
```

It will print B

```
Std.java TapeDeck.java JarHello.java ClientA.java C...\  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = '\u0042';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree:
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student  
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: illegal escape character  
        char degree = '\u0042';
```

```
Student.java:9: error: unclosed character literal  
        char degree = '\u0042';
```

```
Student.java:9: error: unclosed character literal  
        char degree = '\u0042';
```

```
3 errors
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

We can not use invalid escape sequence (like '\U0042' or '\u004')

Example 6 :

```
char degree = '\U0042'
```

Compilation error will be “illegal escape character” and “Unclosed character literal” .

```
testDrive.java TapeDeck.java JarFile.java ClientA.java ClientB.java  
  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree = '\u004';  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + nextId);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: illegal unicode escape  
        char degree = '\u004';  
                                ^  
Student.java:9: error: empty character literal  
        char degree = ''\u004';  
                           ^  
2 errors
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

Example 6 :

```
char degree = '\u004'
```

Compilation error will be “**illegal escape character**” and “**empty character literal**”

```
testDrive.java TapeDeck.java JarFile.java ClientA.java ClientB.java  
  
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
  
    double gpa = 3.8;  
  
    char degree;  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:9: error: illegal unicode escape  
        char degree = '\u004';  
                                ^
```

```
Student.java:9: error: empty character literal  
        char degree = '_\u004';  
                           ^
```

```
2 errors
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

```
id: 1000  
nextId: 1001  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree:   
C:\javaindepth\src\com\semanticsquare\basics>
```

Default value for a char is null (Unicode number = 0), it just doesn't print anything.

Example 6 :
char degree;

Nothing will be printed for class level variable.

```
testDrive.java TapeDeck.java JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java Student.java my.ans Student.java BasicsDemo.java
System.out.println("intHex: " + intHex);

// Java 7
int intBinary = 0b0100_0001;
System.out.println("intBinary: " + intBinary);

int intOctal = 0101;
System.out.println("intOctal: " + intOctal);

char charInt = 65;
System.out.println("charInt: " + charInt);

char charHex = 0x0041;
System.out.println("charHex: " + charHex);

char charBinary = 0b0100_0001;
System.out.println("charBinary: " + charBinary);
}

public static void main(String[] args) {
    // Language Basics 1
    //print();
    primitives();
}
```

Same as integer , Char also supports Hexadecimal , Binary , Octal number base.
Same as integer , Char also supports underscore between digits for readability.

Example 7:

```
char charInt = 65;  
char charHex = 0x0041;  
char charBinary = 0b0100_0001;
```

```
testDrive.java TapeDeck.java JarFile.java ClientA.java ClientB.java  
  
System.out.println("\n");  
int intHex = 0x0041; //  
System.out.println("intHex: " + intHex);  
  
// Java 7  
int intBinary = 0b0100_1000_0000_0001;  
System.out.println("intBinary: " + intBinary);  
  
int intOctal = 0101;  
System.out.println("intOctal: " + intOctal);  
  
char charInt = 65;  
System.out.println("charInt: " + charInt);  
  
char charHex = 0x0041;  
System.out.println("charHex: " + charHex);  
  
char charBinary = 0b0100_1000;  
System.out.println("charBinary: " + charBinary);  
  
}  
  
public static void main(String[] args) {  
    // Language Basics 1  
    //print();  
    primitives();  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside primitives ...
```

```
intHex: 65  
intBinary: 65  
intOctal: 65  
charInt: A  
charHex: A  
charBinary: A
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

They all will be printed as A (equivalent of decimal number 65).

32. Demo: Alternate ways to initialize char variables

```
System.out.println("\n");
int intHex = 0x0041; // A
System.out.println("intHex: " + intHex);

// Java 7
int intBinary = 0b0100_1000_0000_0001;
System.out.println("intBinary: " + intBinary);

int intOctal = 0101;
System.out.println("intOctal: " + intOctal);

char charInt = -1;
System.out.println("charInt: " + charInt);

char charHex = 0x0041;
System.out.println("charHex: " + charHex);

char charBinary = 0b0100_1000_0000_0001;
System.out.println("charBinary: " + charBinary);

}

public static void main(String[] args) {
    // Language Basics 1
    //print();
    primitives();
}
```

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside primitives ...

intHex: 65
intBinary: 65
intOctal: 65
charInt: A
charHex: A
charBinary: A

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

BasicsDemo.java:27: error: incompatible types: possible lossy conversion from int to char

char charInt = -1;

1 error

C:\javaindepth\src\com\semanticsquare\basics>

Example 8:

```
char charInt = -1;
```

It gives us a compilation error, “Incompatible type”, because this is outside of 0 to 65535 and compiler would consider this as Int literal.

```
testDrive.java  File Deck.java  JarHelp.java  Client.java  C...\  
  
System.out.println("\n");  
int intHex = 0x0041; //  
System.out.println("intHex: " + intHex);  
  
// Java 7  
int intBinary = 0b0100_1000_0000_0001;  
System.out.println("intBinary: " + intBinary);  
  
int intOctal = 0101;  
System.out.println("intOctal: " + intOctal);  
  
char charInt = 65536;  
System.out.println("charInt: " + charInt);  
  
char charHex = 0x0041;  
System.out.println("charHex: " + charHex);  
  
char charBinary = 0b0100_1000_0000_0001;  
System.out.println("charBinary: " + charBinary);  
  
}  
  
public static void main(String[] args) {  
    // Language Basics 1  
    //print();  
    primitives();  
}
```

```
C:\javaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaInDepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside primitives ...
```

```
intHex: 65  
intBinary: 65  
intOctal: 65  
charInt: A  
charHex: A  
charBinary: A
```

```
C:\javaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
BasicsDemo.java:27: error: incompatible types: possible lossy conversion from int to char  
        char charInt = -1;
```

```
1 error
```

```
C:\javaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
BasicsDemo.java:27: error: incompatible types: possible lossy conversion from int to char  
        char charInt = 65536;
```

```
1 error
```

```
C:\javaInDepth\src\com\semanticsquare\basics>
```

Example 9:

```
char charInt = 65536;
```

It gives us a compilation error, “Incompatible type”, because this is outside of 0 to 65535 and compiler would consider this as Int literal.

```
testDrive.java TapeDeck.java JarFile.java ClientA.java ClientB.java  
  
System.out.println("\n");  
int intHex = 0x0041; //  
System.out.println("intHex: " + intHex);  
  
// Java 7  
int intBinary = 0b0100_1000_0000_0001;  
System.out.println("intBinary: " + intBinary);  
  
int intOctal = 0101;  
System.out.println("intOctal: " + intOctal);  
  
int intChar = 'A';  
System.out.println("intChar: " + intChar);  
  
char charInt = 65;  
System.out.println("charInt: " + charInt);  
  
char charHex = 0x0041;  
System.out.println("charHex: " + charHex);  
  
char charBinary = 0b0100_1000;  
System.out.println("charBinary: " + charBinary);  
  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
  
Inside primitives ...  
intHex: 65  
intBinary: 65  
intOctal: 65  
intChar: 65  
charInt: A  
charHex: A  
charBinary: A  
C:\javaindepth\src\com\semanticsquare\basics>
```

Let's try opposite conversion

Example 10:

```
Int intChar = 'A';
```

As char and numbers internally taken care as number, it will print 65.

```
System.out.println("\n\nInside primitives ...");
int intHex = 0x0041; // 16 power 0 * 1 + 16 power 1 * 4
System.out.println("intHex: " + intHex);

// Java 7
int intBinary = 0b0100_0001;
System.out.println("intBinary: " + intBinary);

int intOctal = 0101;
System.out.println("intOctal: " + intOctal);

int intChar = 'A';
System.out.println("intChar: " + intChar);

char charInt = 65;
System.out.println("charInt: " + charInt);

char charHex = 0x0041;
System.out.println("charHex: " + charHex);

char charBinary = 0b0100_0001;
System.out.println("charBinary: " + charBinary);

}

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

char variable

unicode escape seq.

char literal

int literal

16-bit unsigned integer

[0,65535]

To summarize,
a char variable can be initialized with
char literal , unicode escape sequence and Int

Boolean Data Type

33. Demo: Primitive Variables: Boolean Data Type

```
class Student {  
    int id = 1000;  
  
    byte age = 18;  
    long phone = 223_456_7890L;  
    double gpa = 3.8;  
    char degree = 'B';  
  
    boolean international = true;  
  
    void compute() {  
        int nextId = id + 1;  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
        System.out.println("degree: " + degree);  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.compute();  
    }  
}
```

Boolean data type, take either a true or false value.

Default, value for Boolean is false.

```
va JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini Student.java Student1.java new_3 new_4 new_5 new_6
class Student {
    int id = 1000;

    byte age = 18;
    long phone = 223_456_7890L;
    double gpa = 3.8;
    char degree = 'B';

    boolean international = true;
    double tuitionFees = 12000.0;
    double internationalFees = 5000.0;

    void compute() {
        int nextId = id + 1;

        if (international == true) {
            tuitionFees = tuitionFees + internationalFees;
        }

        System.out.println("id: " + id);
        System.out.println("nextId: " + nextId);
        System.out.println("age: " + age);
        System.out.println("phone: " + phone);
        System.out.println("gpa: " + gpa);
        System.out.println("degree: " + degree);
    }
}
```

Example :

If the student is a local student, then student would pay a certain fees.

But if the student is an international student, then he or she would have to pay higher fee.

```
boolean international = true;  
double tuitionFees = 12000.0;  
double internationalFees = 5000.0;  
  
if (international == true)  
{  
    tuitionFees = tuitionFees + internationalFees;  
}  
  
System.out.println("TuitionFees :" + tuitionFees)
```

If international is equal to true,

Tuition fees equal to tuition fees plus international fees (17000).

Else

tuition fees will be unchanged.

33. Demo: Primitive Variables: Boolean Data Type

```
boolean international = true;
double tuitionFees = 12000.0;
double internationalFees = 5000.0;

void compute() {
    int nextId = id + 1;

    if (international == true)
        tuitionFees = tuitionFees * 1.1;
}

System.out.println("id: " + id);
System.out.println("nextId: " + nextId);
System.out.println("age: " + age);
System.out.println("phone: " + phone);
System.out.println("gpa: " + gpa);
System.out.println("degree: " + degree);
System.out.println("tuitionFees: " + tuitionFees);

}

public static void main(String[] args) {
    Student s = new Student();
    s.compute();
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 17000.0
```

Time to compile and execute.

It prints 17000.

```
va JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini Student.java Student1.java new_3 new_4 new_5 new_6
boolean international = true;
double tuitionFees = 12000.0;
double internationalFees = 5000.0;

void compute() {
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("id: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
}

public static void main(String[] args) {
    Student s = new Student();
    s.compute();
}
```

We don't have to use equal operator explicitly.

Example :

If (**international** == **true**)

can be written as

If (**international**)



```
boolean international;           |
double tuitionFees = 12000.0;
double internationalFees = 5000.0;

void compute() {
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("id: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
}

public static void main(String[] args) {
    Student s = new Student();
    s.compute();
```

Example :

```
boolean international;
```

This will assign false for international variable at class level because Boolean has default value as false.

```
java JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini new_7 Student.java new_3 new_4 new_5 new_6
```

```
class Student {  
    int id = 1000;  
    String name = "John";  
    String gender = "male";  
    int age = 18;  
    long phone = 223_456_7890L;  
    double gpa = 3.8;  
    char degree = 'B';  
  
    boolean international;  
    double tuitionFees = 12000.0;  
    double internationalFees = 5000.0;  
  
    void compute() {  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
  
        System.out.println("id: " + id);  
        System.out.println("nextId: " + nextId);  
        System.out.println("age: " + age);  
        System.out.println("phone: " + phone);  
        System.out.println("gpa: " + gpa);  
    }  
}
```

We looked at the 8 primitive types, they are inbuilt into the language.

Variables can be primitive variables, they can also be object references, with object references, the type of the variable will be a Class or an Interface.

There are 3 kinds of variables.

Instance , Static(Class) and Local Variable

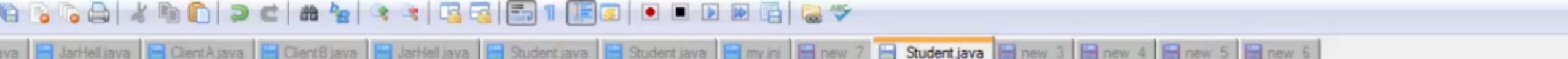
34. Demo: Variable Kinds

```
if (international) {  
    tuitionFees = tuitionFees + internationalFees;  
}  
  
System.out.println("id: " + id);  
System.out.println("nextId: " + nextId);  
System.out.println("age: " + age);  
System.out.println("phone: " + phone);  
System.out.println("gpa: " + gpa);  
System.out.println("degree: " + degree);  
System.out.println("tuitionFees: " + tuitionFees);  
}
```

```
public static void main(String[] args) {  
    Student student1 = new Student();  
    student1.id = 1000;  
    student1.name = "John";  
    student1.gender = "male";  
    student1.age = 18;  
    student1.phone = 223_456_7890L;  
    student1.gpa = 3.8;  
    student1.degree = 'B';  
      
    student1.international = false;
```

Instance variables :

- Variables which holds state of object are nothing but the instance variables.
- They are declared at class level without static keyword.
- If no value assign , they gets default value.



- From outside class: Accessible via Class Name or object reference

Instance & static variables are also referred to as fields or attributes. Attributes is probably more commonly associated with instance variables.

*/

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 12000.0;  
    double internationalFees = 5000.0;  
  
    void compute() {  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
    }  
}
```

Static/Class variable :

- Variables with static keyword.
- They are declared at class level with static keyword.
- It maintains a single copy of the variable regardless they were changed by any instance/object or not.
- They are specific to the class.
- If no value assign , they gets default value.

34. Demo: Variable Kinds

```
double internationalFees = 5000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
    System.out.println("computeCount: " + computeCount);
}

public static void main(String[] args) {
    Student student1 = new Student();
}
```

Local variables :

- Local variables are declared at Method level, they cannot be accessed outside of the method.
- If no value assign , they will give compilation error (they does not get default value).

Java JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini new_7 Student.java new_3 new_4 new_5 new_6

Instance & static variables are also referred to as fields or attributes. Attributes is probably more commonly associated with instance variables.

```
/*
class Student {
    int id;
    String name;
    String gender;
    int age;
    long phone;
    double gpa;
    char degree;

    boolean international;
    double tuitionFees = 12000.0;
    double internationalFees = 5000.0;

    void compute() {
        int nextId = id + 1;

        if (international) {
            tuitionFees = tuitionFees + internationalFees;
        }

        System.out.println("id: " + id);
    }
}
```

© semanticsquare.com 63 Ln: 30 Col: 28 Sel: 0 Dos\Windows ANSI

Let's check the instance variables

Example 1:

```
int id;  
String name;  
String gender;  
Int age;  
long phone;  
double gpa;  
char degree;  
boolean international;  
double tuitionFees = 12000.0;  
double internationalFees = 5000.0;
```

34. Demo: Variable Kinds

```
double gpa;
char degree;

boolean international;
double tuitionFees = 12000.0;
double internationalFees = 5000.0;

void compute() {
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("\nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);           I
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);

}
```

Let's print them using compute method.

```
}
```

```
public static void main(String[] args) {
    Student student1 = new Student();
    student1.id = 1000;
    student1.name = "John";
    student1.gender = "male";
    student1.age = 18;
    student1.phone = 223_456_7890L;
    student1.gpa = 3.8;
    student1.degree = 'B';
    student1.international = false;
    student1.compute();
```

Let's create first object student1 and set the state of it using instance variables.

```
va JarHell.java ClientA.java ClientB.java JarHell.java

void compute() {
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuiti
    }

    System.out.println("ni
    System.out.println("nex
    System.out.println("nam
    System.out.println("gen
    System.out.println("age
    System.out.println("pho
    System.out.println("gpa
    System.out.println("deg
    System.out.println("tui

}

public static void main(Str
    Student student1 = new
    student1.id = 1000;
    student1.name = "John";
    student1.gender = "male
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

```
id: 1000
nextId: 1001
name: John
gender: male
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 12000.0
```

```
C:\javaindepth\src\com\semanticsquare\basics>_
```

They all will compile and execute fine.

```
public static void main(String[] args) {  
    Student student1 = new Student();  
    student1.id = 1000;  
    student1.name = "John";  
    student1.gender = "male";  
    student1.age = 18;  
    student1.phone = 223_456_7890L;  
    student1.gpa = 3.8;  
    student1.degree = 'B';  
    student1.international = false;  
    student1.compute();  
  
    Student student2 = new Student();  
    student2.id = 1001;  
    student2.name = "Raj";  
    student2.gender = "male";  
    student2.age = 21;  
    student2.phone = 223_456_9999L;  
    student2.gpa = 3.4;  
    student2.degree = 'M';  
    student2.international = true;  
    student2.compute();  
}
```

Similarly let's create second object student2 and set the state of it using instance variables.

```
java JarHell.java ClientA.java ClientB.java JarHell.java

}

public static void main(Str
    Student student1 = new
student1.id = 1000;
student1.name = "John";
student1.gender = "male"
student1.age = 18;
student1.phone = 223_45
student1.gpa = 3.8;
student1.degree = 'B';
student1.international
student1.compute();

Student student2 = new
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male"
student2.age = 21;
student2.phone = 223_45
student2.gpa = 3.4;
student2.degree = 'M';
student2.international
student2.compute();
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java

C:\javaindepth\src\com\semanticsquare\basics>java Student

id: 1000
nextId: 1001
name: John
gender: male
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 12000.0

id: 1001
nextId: 1002
name: Raj
gender: male
age: 21
phone: 2234569999
gpa: 3.4
degree: M
tuitionFees: 17000.0
```

This will also run fine.

```
java JarHell.java ClientA.java ClientB.java JarHell.java

student1.degree = 'B';
student1.international
student1.compute();

Student student2 = new
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male"
student2.age = 21;
student2.phone = 223_45
student2.gpa = 3.4;
student2.degree = 'M';
student2.international
student2.compute();

Student student3 = new
student3.id = 1002;
student3.name = "Anita"
student3.gender = "fema"
student3.age = 20;
student3.phone = 223_45
student3.gpa = 4.0;
student3.degree = 'M';
student3.international
student3.compute();
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
name: John
gender: male
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 12000.0

id: 1001
nextId: 1002
name: Raj
gender: male
age: 21
phone: 2234569999
gpa: 3.4
degree: M
tuitionFees: 17000.0

id: 1002
nextId: 1003
name: Anita
gender: female
age: 20
phone: 2234568888
gpa: 4.0
degree: M
tuitionFees: 17000.0

C:\javaindepth\src\com\semanticsquare\basics>
```

Similarly instance variable for 3rd object will also work.

```
/*  
  
class Student {  
    int computeCount;  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 12000.0;  
    double internationalFees = 5000.0;  
  
    void compute() {  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
  
        System.out.println("\nid: " + id);  
        System.out.println("nextId: " + nextId);  
    }  
}
```

Now, let's look at the Static or Class variables.

Example 2 :

let's say we want to keep track of the number of times compute method is invoked.

So we can declare an instance variable below:

```
Int computeCount;
```

```
java JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini new_7 Student.java new_3 new_4 new_5 new_6

boolean international;
double tuitionFees = 12000.0;
double internationalFees = 5000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("\nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
}

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

We will increase computeCount within the compute each time the method invoked

```
computeCount = computeCount + 1;
```

However whenever new object will use Count method, the computeCount will reset to it's original value (0 in this case).

To keep computeCount independent from objects/instances, we have to convert computeCount to Class/Static variable.

```
java JarHello.java ClientA.java ClientB.java JarHello.java  
- From outside class: Access  
Instance & static variables are more commonly associated with  
*/  
  
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 12000.  
    double internationalFees =  
  
    void compute() {  
        computeCount = computeC  
        int nextId = id + 1;
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

```
id: 1000  
nextId: 1001  
name: John  
gender: male  
age: 18  
phone: 2234567890  
gpa: 3.8  
degree: B  
tuitionFees: 12000.0  
computeCount: 1
```

```
id: 1001  
nextId: 1002  
name: Raj  
gender: male  
age: 21  
phone: 2234569999  
gpa: 3.4  
degree: M  
tuitionFees: 17000.0  
computeCount: 2
```

```
id: 1002  
nextId: 1003  
name: Anita  
gender: female  
age: 20  
phone: 2234568888  
gpa: 4.0  
degree: M  
tuitionFees: 17000.0  
computeCount: 3
```

Now with static :

```
static int computeCount;
```

We will get increase count of computeCount method even if it has been invoked by object/
instance.

```
student1.compute();

Student student2 = new Student();
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male";
student2.age = 21;
student2.phone = 223_456_9999L;
student2.gpa = 3.4;
student2.degree = 'M';
student2.international = true;
student2.compute();

Student student3 = new Student();
student3.id = 1002;
student3.name = "Anita";
student3.gender = "female";
student3.age = 20;
student3.phone = 223_456_8888L;
student3.gpa = 4.0;
student3.degree = 'M';
student3.international = true;
student3.compute();

System.out.println("Student.computeCount: " + Student.computeCount);
```

Now if we want to access the class variable from same or other class , it's recommended to access it via Class name

Example 1:

```
System.out.println("Student.computeCount : " +  
Student.computeCount);
```

```
va JarHell.java ClientA.java ClientB.java JarHell.java
Student student2 = new
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male";
student2.age = 21;
student2.phone = 223_45;
student2.gpa = 3.4;
student2.degree = 'M';
student2.international
student2.compute();

Student student3 = new
student3.id = 1002;
student3.name = "Anita";
student3.gender = "fema
student3.age = 20;
student3.phone = 223_45;
student3.gpa = 4.0;
student3.degree = 'M';
student3.international
student3.compute();

System.out.println("Stu
}

}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
C:\javaindepth\src\com\semanticsquare\basics>java Student
id: 1000
nextId: 1001
name: John
gender: male
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 12000.0
computeCount: 1

id: 1001
nextId: 1002
name: Raj
gender: male
age: 21
phone: 2234569999
gpa: 3.4
degree: M
tuitionFees: 17000.0
computeCount: 2

id: 1002
nextId: 1003
name: Anita
gender: female
age: 20
phone: 2234568888
gpa: 4.0
degree: M
tuitionFees: 17000.0
computeCount: 3
Student.computeCount: 3
```

This will execute fine.

```
Student student2 = new Student();
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male";
student2.age = 21;
student2.phone = 223_456_9999L;
student2.gpa = 3.4;
student2.degree = 'M';
student2.international = true;
student2.compute();

Student student3 = new Student();
student3.id = 1002;
student3.name = "Anita";
student3.gender = "female";
student3.age = 20;
student3.phone = 223_456_8888L;
student3.gpa = 4.0;
student3.degree = 'M';
Student.international = true;
student3.compute();

System.out.println("Student.computeCount: " + Student.computeCount);
}
```

However the reserve of it (access instance variable using Class name) would not work.

Student.international = true;

will give a compilation error

/*

Instance & Static:

- Declared at class-level
- Scope: Entire class
- Gets default value
- Cannot be re-initialized directly within class

Instance: Represents object state

- Values are unique to object
- From outside class: Accessible via object reference

Static:

- Values are unique to class ~ One copy per class (shared across objects)
- From outside class: Accessible via Class Name or object reference

Instance & static variables are also referred to as fields or attributes. Attributes is probably more commonly associated with instance variables.

*/

```
class Student {  
    static int computeCount;  
  
    int id;
```



Here are some notes about Instance and Static variables.

34. Demo: Variable Kinds

```
double internationalFees = 5000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
    System.out.println("computeCount: " + computeCount);
}

public static void main(String[] args) {
    Student student1 = new Student();
}
```

Local Variable

local variables are declared within method or as method parameters.

They don't get any default value during declaration.

Their scope is limited to method.

34. Demo: Variable Kinds

```
double internationalFees = 5000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("\nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
    System.out.println("computeCount: " + computeCount);
}

public static void main(String[] args) {
    Student student1 = new Student();
}
```

Local variable declared under method utilized under same method only

Example 1:

```
Int nextId = id + 1;  
  
.  
.  
System.out.println("nextId: "+ nextId);
```

```
public static void main(String[] args) {  
    Student student1 = new Student();  
    student1.id = 1000;  
    student1.name = "John";  
    student1.gender = "male";  
    student1.age = 18;  
    student1.phone = 223_456_7890L;  
    student1.gpa = 3.8;  
    student1.degree = 'B';  
    student1.international = false;  
    student1.compute();  
  
    Student student2 = new Student();  
    student2.id = 1001;  
    student2.name = "Raj";  
    student2.gender = "male";  
    student2.age = 21;  
    student2.phone = 223_456_9999L;  
    student2.gpa = 3.4;  
    student2.degree = 'M';  
    student2.international = true;  
    student2.compute();  
}
```

Local variable declared under method parenthesis.

Example 2:

```
public static void main(String[] args)
{
...
}
```

```
java JarHell.java ClientA.java ClientB.java JarHell.java Student.java Student.java my.ini new_7 Student.java new_3 new_4 new_5 new_6

Student student2 = new Student();
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male";
student2.age = 21;
student2.phone = 223_456_9999L;
student2.gpa = 3.4;
student2.degree = 'M';
student2.international = true;
student2.compute();

Student student3 = new Student();
student3.id = 1002;
student3.name = "Anita";
student3.gender = "female";
student3.age = 20;
student3.phone = 223_456_8888L;
student3.gpa = 4.0;
student3.degree = 'M';
student3.international = true;
student3.compute();

System.out.println("Student.computeCount: " + student3.nextId);

}
```

Local variable can not be accessed outside the method where it has been declared.

Example 3:

```
class Student{  
  
    void compute ()  
    {  
        Int nextId = id + 1;  
        .  
        .  
    }  
  
    System.out.println(student3.nextId);  
  
}
```

The above code will give the compilation error.

```
va JarHell.java ClientA.java ClientB.java JarHell.java
Student student2 = new
student2.id = 1001;
student2.name = "Raj";
student2.gender = "male";
student2.age = 21;
student2.phone = 223_45;
student2.gpa = 3.4;
student2.degree = 'M';
student2.international;
student2.compute();

Student student3 = new
student3.id = 1002;
student3.name = "Anita";
student3.gender = "fema";
student3.age = 20;
student3.phone = 223_45;
student3.gpa = 4.0;
student3.degree = 'M';
student3.international;
student3.compute();

System.out.println("Stu
}
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
Student.java:91: error: cannot find symbol
                                System.out.println("Student.computeCount: " + student3.nextId)
                                         ^
symbol:   variable nextId
location: variable student3 of type Student
1 error
```

```
C:\javaindepth\src\com\semanticsquare\basics>_
```

The above code will give the compilation error "**Cannot find symbol**".

A screenshot of a Java IDE interface. The top bar shows various icons and tabs, with the 'Student.java' tab currently selected. Below the tabs, a list of files is visible, including 'JarHell.java', 'ClientA.java', 'ClientB.java', 'JarHell.java', 'Student.java', 'my.ini', 'new_7', 'Student.java', 'new_3', 'new_4', 'new_5', and 'new_6'. The main workspace displays Java code for a 'Student' class. The code includes a static variable 'tuitionFees', instance variables 'international', 'id', 'name', 'gender', 'age', 'phone', 'gpa', 'degree', and 'computeCount', and a 'compute' method that prints out these variables and their values.

```
double tuitionFees = 12000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    double internationalFees;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("\nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
    System.out.println("computeCount: " + computeCount);
}
```

Local variable does not get default value during declaration.

Example 4:

```
double internationalFees;
```

The above local variable will give compilation error.

34. Demo: Variable Kinds

```
double tuitionFees = 12000.  
  
void compute() {  
    computeCount = computeC  
    int nextId = id + 1;  
  
    double internationalFee  
  
    if (international) {  
        tuitionFees = tuiti  
    }  
  
    System.out.println("\ni  
    System.out.println("nex  
    System.out.println("nam  
    System.out.println("gen  
    System.out.println("age  
    System.out.println("pho  
    System.out.println("gpa  
    System.out.println("deg  
    System.out.println("tui  
    System.out.println("com  
  
}
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:91: error: cannot find symbol  
                                System.out.println("Student.computeCount: " + student3.nextId)  
                                ^  
                                     symbol:   variable nextId  
                                     location: variable student3 of type Student  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:43: error: variable internationalFees might not have been initialized  
                                tuitionFees = tuitionFees + internationalFees;  
                                ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>_
```



```
double tuitionFees = 12000.0;

void compute() {
    computeCount = computeCount + 1;
    int nextId = id + 1;

    double internationalFees;
    internationalFees = 5000.0;

    if (international) {
        tuitionFees = tuitionFees + internationalFees;
    }

    System.out.println("nid: " + id);
    System.out.println("nextId: " + nextId);
    System.out.println("name: " + name);
    System.out.println("gender: " + gender);
    System.out.println("age: " + age);
    System.out.println("phone: " + phone);
    System.out.println("gpa: " + gpa);
    System.out.println("degree: " + degree);
    System.out.println("tuitionFees: " + tuitionFees);
    System.out.println("computeCount: " + computeCount);
}
```

Example 5:

```
double internationalFees;  
InternationalFees = 5000.0;
```

Local variable does not get default value during declaration and give compilation error even if we assign the value to Local variable later.

34. Demo: Variable Kinds

```
double tuitionFees = 12000.  
  
void compute() {  
    computeCount = computeC  
    int nextId = id + 1;  
  
    double internationalFee  
    internationalFees = 500  
  
    if (international) {  
        tuitionFees = tuiti  
    }  
  
    System.out.println("ni  
    System.out.println("nex  
    System.out.println("nam  
    System.out.println("gen  
    System.out.println("age  
    System.out.println("pho  
    System.out.println("gpa  
    System.out.println("deg  
    System.out.println("tui  
    System.out.println("com
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:91: error: cannot find symbol  
                                System.out.println("Student.computeCount: " + student3.nextId)  
                                ^  
                                     symbol:   variable nextId  
                                     location: variable student3 of type Student  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
Student.java:43: error: variable internationalFees might not have been initialized  
                                tuitionFees = tuitionFees + internationalFees;  
                                ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
C:\javaindepth\src\com\semanticsquare\basics>
```


Multi Variable Declaration Statement

35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 12000.0, internationalFees = 5000.0;  
}  
  
void compute() {  
    computeCount = computeCount + 1;  
    int nextId = id + 1;  
  
    if (international) {  
        tuitionFees = tuitionFees + internationalFees;  
    }  
  
    System.out.println("\nid: " + id);  
    System.out.println("nextId: " + nextId);  
}
```

- It perfectly valid to have a declaration statement declaring multiple variables of same data type.
- We can also initialize the variables in the same statement.

Example 1 :

```
double tuitionFees = 12000.0, internationalFees = 5000.0;
```

```
.
```

```
.
```

```
.
```

```
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will run fine

35. Demo: Multi-variable Declaration Statements

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
class Student {  
    int id;  
    String name;  
    String gender;  
    int age;  
    String phone;  
    float gpa;  
    String degree;  
    int tuitionFees;  
    long computeCount;  
  
    double id; // This is a compile error.  
    char name; // This is a compile error.  
    gender: male; // This is a compile error.  
    age: 21; // This is a compile error.  
    phone: 2234567890; // This is a compile error.  
    gpa: 3.8; // This is a compile error.  
    degree: B; // This is a compile error.  
    tuitionFees: 12000.0; // This is a compile error.  
    computeCount: 1; // This is a compile error.  
  
    void id: 1001; // This is a compile error.  
    void nextId: 1002; // This is a compile error.  
    void name: Raj; // This is a compile error.  
    void gender: male; // This is a compile error.  
    void age: 21; // This is a compile error.  
    void phone: 2234567890; // This is a compile error.  
    void gpa: 3.8; // This is a compile error.  
    void degree: B; // This is a compile error.  
    void tuitionFees: 12000.0; // This is a compile error.  
    void computeCount: 1; // This is a compile error.  
  
    void id: 1002; // This is a compile error.  
    void nextId: 1003; // This is a compile error.  
    void name: Anita; // This is a compile error.  
    void gender: female; // This is a compile error.  
    void age: 20; // This is a compile error.  
    void phone: 2234568888; // This is a compile error.  
    void gpa: 4.0; // This is a compile error.  
    void degree: M; // This is a compile error.  
    void tuitionFees: 17000.0; // This is a compile error.  
    void computeCount: 2; // This is a compile error.  
  
    void Student.computeCount: 3; // This is a compile error.
```




Client8.java JarHell.java myuni new_7 new_6 Student.java

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 12000.0, double internationalFees = 5000.0;  
  
    void compute() {  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
  
        System.out.println("\nid: " + id);  
        System.out.println("nextId: " + nextId);  
    }  
}
```

As multiple variable declaration is possible of same data type, we can not even mention additional data type in the statement.

Example 3 :

```
double tuitionFees = 12000.0, double internationalFees = 5000.0;  
. . .  
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will give compilation error “**identifier expected**”

35. Demo: Multi-variable Declaration Statements

```
Student.java:14: error: <identifier> expected
    double tuitionFees = 12000.0, double internationalFees = 5000.0;
                                         ^
1 error
```

C:\javaindepth\src\com\semanticsquare\basics>

< >

|| 1x 1:47 / 9:31

© semanticsquare.com

35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees, internationalFees = 5000.0;  
  
    void compute() {  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
  
        System.out.println("\nid: " + id);  
        system.out.println(nextId: " + nextId);  
    }  
}
```

Example 1 :

```
double tuitionFees, internationalFees = 5000.0;  
. . .  
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will assign default value to tuitionFees (for instance and static variable) and
internationalFees = 5000.0

35. Demo: Multi-variable Declaration Statements

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
class Student {  
    int id: 1000;  
    static int nextId: 1001;  
    String name: John;  
    String gender: male;  
    int age: 18;  
    String phone: 2234567890;  
    String gpa: 3.8;  
    String degree: B;  
    int tuitionFees: 0.0;  
    long computeCount: 1;  
    double id: 1001;  
    char nextId: 1002;  
    char name: Raj;  
    char gender: male;  
    int age: 21;  
    String phone: 2234569999;  
    float gpa: 3.4;  
    String degree: M;  
    float tuitionFees: 5000.0;  
    void computeCount: 2;  
  
    int id: 1002;  
    int nextId: 1003;  
    String name: Anita;  
    String gender: female;  
    int age: 20;  
    String phone: 2234568888;  
    float gpa: 4.0;  
    String degree: M;  
    float tuitionFees: 5000.0;  
    long computeCount: 3;  
    Student.computeCount: 3;
```

C:\javaindepth\src\com\semanticsquare\basics>

35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
  
    void compute() {  
        I  
        double tuitionFees, internationalFees = 5000.0;  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            | tuitionFees = tuitionFees + internationalFees;  
        }  
  
        system.out.println( \nid:  + id);  
    }  
}
```

If the below declaration will be used under method then , we will get compilation error “**variable might not have initialized**”. Because tuitionFees will not get default value.

Example 2 :

```
double tuitionFees, internationalFees = 5000.0;
```

```
.
```

```
.
```

```
System.out.println("tuitionFees : " + tuitionFees);
```

35. Demo: Multi-variable Declaration Statements

```
\basics>javac Student.java
Student.java:23: error: variable tuitionFees might not have been initialized
                      tuitionFees = tuitionFees + internationalFees;
                                         ^
Student.java:34: error: variable tuitionFees might not have been initialized
                      System.out.println("tuitionFees: " + tuitionFees);
                                         ^
2 errors
```

```
C:\javaindepth\src\com\semanticsquare\basics>_
```

```
String
String
int
long
double
char
```

```
boolean
```

```
void
```



1x 3:22 / 9:31



35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 100, internationalFees = tuitionFees + 5000.0;  
  
    void compute() {  
  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees + internationalFees;  
        }  
    }  
}
```

14 people have written a note here.

Example 3 :

```
double tuitionFees = 100, internationalFees = tuitionFees + 5000.0;  
. . .  
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will work without any issue.

35. Demo: Multi-variable Declaration Statements

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java Student
```

```
int id: 1000
String nextId: 1001
String name: John
String gender: male
int age: 18
long phone: 2234567890
double gpa: 3.8
double degree: B
char tuitionFees: 100.0
char computeCount: 1
```

```
boolean id: 1001
nextId: 1002
double name: Raj
gender: male
age: 21
void phone: 2234569999
gpa: 3.4
degree: M
tuitionFees: 5200.0
computeCount: 2
```

```
id: 1002
nextId: 1003
name: Anita
gender: female
```

14 people have written a note here.

```
gpa: 3.4
degree: M
```

```
tuitionFees: 5200.0
```

```
computeCount: 3
```


35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    double tuitionFees = 100.0;  
    double internationalFees = tuitionFees = 5000.0;  
  
    void compute() {  
  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
    }  
}
```

For Instance/static variable , usually we can not re initialize them at class level.
However if the re-initialization done as part of declaration statement , it will work without any issue.

Example 4 :

```
double tuitionFees = 100,  
double internationalFees = tuitionFees = 5000.0;  
. . .  
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will work without any issue.

35. Demo: Multi-variable Declaration Statements

```
C:\javaindepth\src\com\semanticsquare\basics>javac Student.java  
  
class Student {  
    int id;  
    String name;  
    String gender;  
    int age;  
    String phone;  
    double gpa;  
    String degree;  
    int tuitionFees;  
    long computeCount;  
  
    double id; // This is a local variable  
    double nextId; // This is a local variable  
    String name; // This is a local variable  
    String gender; // This is a local variable  
    int age; // This is a local variable  
    String phone; // This is a local variable  
    double gpa; // This is a local variable  
    String degree; // This is a local variable  
    int tuitionFees; // This is a local variable  
    long computeCount; // This is a local variable  
  
    void id(int id) {  
        this.id = id;  
        System.out.println("id: " + id);  
    }  
  
    void name(String name) {  
        this.name = name;  
        System.out.println("name: " + name);  
    }  
  
    void gender(String gender) {  
        this.gender = gender;  
        System.out.println("gender: " + gender);  
    }  
  
    void age(int age) {  
        this.age = age;  
        System.out.println("age: " + age);  
    }  
  
    void phone(String phone) {  
        this.phone = phone;  
        System.out.println("phone: " + phone);  
    }  
  
    void gpa(double gpa) {  
        this.gpa = gpa;  
        System.out.println("gpa: " + gpa);  
    }  
  
    void degree(String degree) {  
        this.degree = degree;  
        System.out.println("degree: " + degree);  
    }  
  
    void tuitionFees(int tuitionFees) {  
        this.tuitionFees = tuitionFees;  
        System.out.println("tuitionFees: " + tuitionFees);  
    }  
  
    void computeCount(long computeCount) {  
        this.computeCount = computeCount;  
        System.out.println("computeCount: " + computeCount);  
    }  
  
    void Student.computeCount() {  
        System.out.println("Student.computeCount: " + computeCount);  
    }  
}
```


35. Demo: Multi-variable Declaration Statements

```
class Student {  
    static int computeCount;  
  
    int id;  
    String name;  
    String gender;  
    int age;  
    long phone;  
    double gpa;  
    char degree;  
  
    boolean international;  
    //double tuitionFees = 100.0;  
    double internationalFees = 5000.0;  
  
    void compute() {  
  
        computeCount = computeCount + 1;  
        int nextId = id + 1;  
  
        if (international) {  
            tuitionFees = tuitionFees + internationalFees;  
        }  
    }  
}
```

Example 5 :

```
double internationalFees = tuitionFees = 5000.0;
```

```
.
```

```
.
```

```
System.out.println("tuitionFees : " + tuitionFees);
```

The above code will give compilation error “**can not find symbol**”.

35. Demo: Multi-variable Declaration Statements

```
Student.java:15: error: cannot find symbol
    double internationalFees = tuitionFees = 5000.0;
                           ^
      symbol:  variable tuitionFees
      location: class Student
Student.java:24: error: cannot find symbol
                tuitionFees = tuitionFees + internationalFees;
                           ^
      symbol:  variable tuitionFees
      location: class Student
Student.java:24: error: cannot find symbol
                tuitionFees = tuitionFees + internationalFees;
                           ^
      symbol:  variable tuitionFees
      location: class Student
Student.java:35: error: cannot find symbol
        System.out.println("tuitionFees: " + tuitionFees);
                           ^
      symbol:  variable tuitionFees
      location: class Student
//4 errors
```

C:\javaindepth\src\com\semanticsquare\basics>_

Variables: Type Casting

Dheeru Mundluru



Type Casting

Assign variable or literal of one type to variable of another type

- int ← long
- int ← byte
- ▶ Only **numeric-to-numeric** casting is possible
- ▶ Cannot cast to **boolean** or vice versa
- ▶ Implicit or explicit

Java is statically type language, however sometimes we may have to **assign value of one data type to other data type**, this is called **Typecasting**.

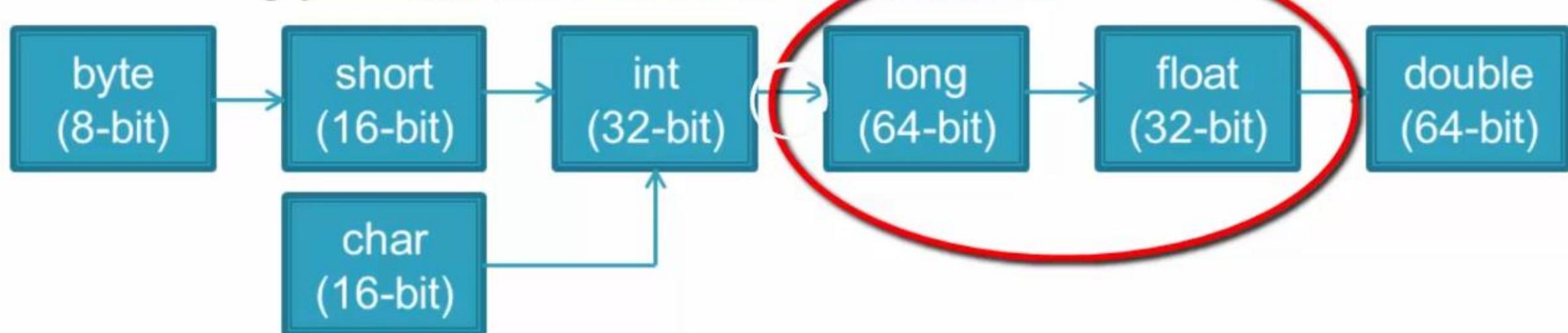
- Typecasting is possible only between numeric primitive type (primitive type other than Boolean).
- Typecasting can be either implicit or explicit.

Implicit Casting

- Smaller to larger ~ *widening* conversion

```
int x = 65;
```

```
long y = x; (Implicit casting by compiler)
```



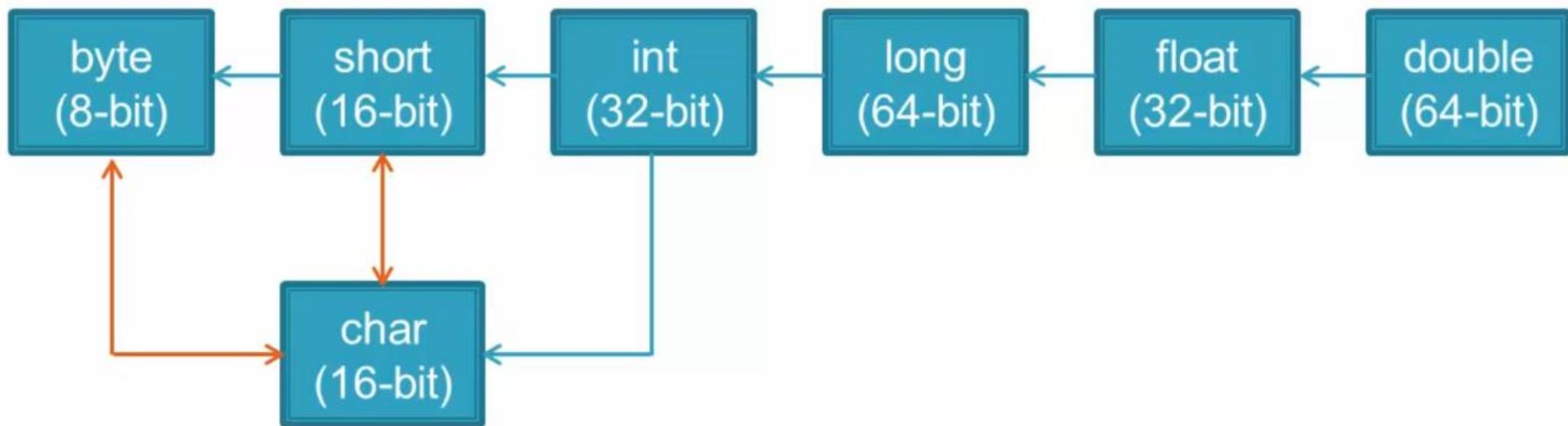
- Integer to Floating-point is *implicit* too

Implicit casting (widening conversion) , casting variable or literal from smaller data type to larger data type.

- Smaller and larger types is based on the range of values the data type supports.
- Implicit casting is performed by the compiler automatically
- The diagram shows the order of implicit casting performed by Compiler

Explicit Casting

Larger to smaller ~ *narrowing conversion*



Explicit casting (narrowing conversion) , casting variable or literal from larger data type to smaller data type.

- Explicit casting is performed by the programmer.
- The diagram shows the order of explicit casting performed by Compiler .
- Any assignments between byte <-> char and short <-> char would require an Explicit cast.
- The main reason for explicit casting is because the value assigned could be outside the range of the variable on the left hand side.

36. Variables: Type Casting + Demo

```
static void typeCasting() {  
    System.out.println("\nInside typeCasting ...");  
    // Explicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to out-of-range assignment  
    byte narrowdByte = (byte)123456;  
    System.out.println("narrowdByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.99;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to long)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // special conversion (widening from byte --> int followed by
```

Let's take a look on few examples.

```
static void typeCasting() {  
    System.out.println("\n\nExplicit casting  
    long y = 42;  
    int x = y;  
    //int x = (int)y;  
  
    // Information loss due to  
    byte narrowdByte = (byte)  
    System.out.println("narrowedByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to char)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // same result as above  
}
```

C:\Windows\system32\cmd.exe
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:43: error: incompatible types: possible lossy conversion from long to int
 int x = y;
 ^
1 error
C:\javaindepth\src\com\semanticsquare\basics>

Without Explicit casting

Example 1 :

```
long y = 42;  
Int x = y;
```

Even though 42 is a valid value for int. The Compiler does not care about it, it sees long to int conversion hence expect us to mention the same.

Hence , we will get the compilation error “**Incompatible type: possible lossy conversion**”

```
static void typeCasting() {  
    System.out.println("\n\nExplicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to  
    byte narrowdByte = (byte)  
    System.out.println("narrowedByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to char)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // same as (char)(byte)bByte;  
    System.out.println("cChar: " + cChar);  
}  
  
C:\Windows\system32\cmd.exe  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:43: error: incompatible types: possible lossy conversion from long to int  
        int x = y;  
                           ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>
```

With Explicit casting

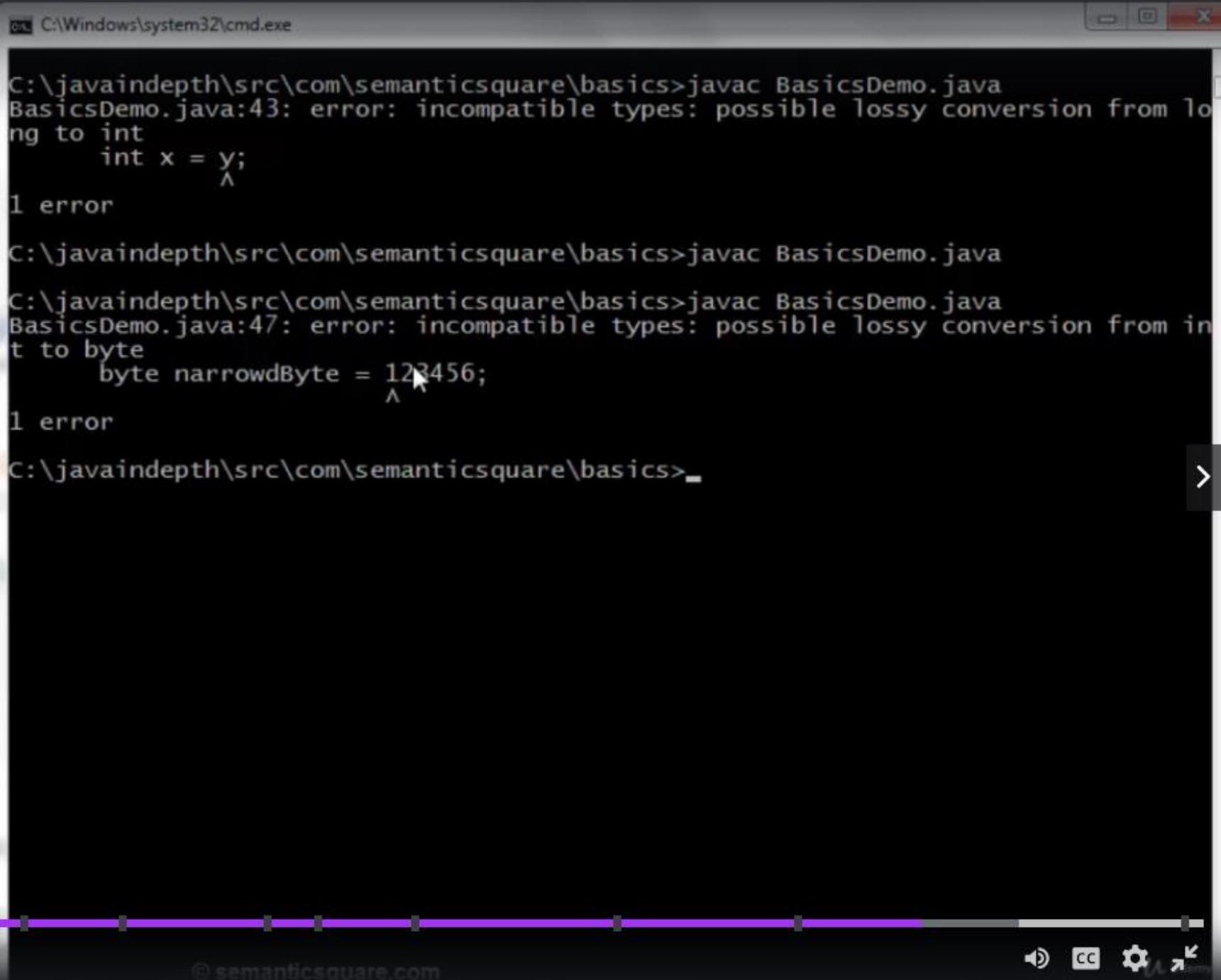
Example 2 :

```
long y = 42;  
Int x = (int) y;
```

This will run fine as we have mentioned the explicit cast (**int**)

36. Variables: Type Casting + Demo

```
static void typeCasting() {  
    System.out.println("\n\nExplicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to  
    byte narrowdByte = 123456;  
    System.out.println("narrowdByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to char)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // same as (char)(byte)bByte
```



The screenshot shows a Windows Command Prompt window titled 'cmd.exe' with the path 'C:\JavaInDepth\src\com\semanticsquare\basics'. The window displays three separate javac command outputs:

- The first output shows an error for explicit casting from long to int: 'BasicsDemo.java:43: error: incompatible types: possible lossy conversion from long to int'. The cursor is positioned at the '^' character in the line 'int x = y;'.
- The second output shows an error for narrowing from int to byte: 'BasicsDemo.java:47: error: incompatible types: possible lossy conversion from int to byte'. The cursor is positioned at the '^' character in the line 'byte narrowdByte = 123456;'.
- The third output shows an error for implicit casting from char to int: 'BasicsDemo.java:51: error: incompatible types: possible lossy conversion from char to int'. The cursor is positioned at the '^' character in the line 'int iInt = cChar;'.

At the bottom of the window, there are standard operating system status icons and a timestamp '16:20 / 19:25'.

Example 3:

```
byte narrowdByte = 123456;  
  
System.out.println("narrowdByte: " + narrowdByte);
```

Since 123456 (Int literal) is out of byte range it will give compilation error “Incompatible Type:
possible lossy conversion”

36. Variables: Type Casting + Demo

```
static void typeCasting() {  
    System.out.println("\n\nExplicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to  
    byte narrowdByte = (byte)  
    System.out.println("narrowedByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to  
    y = x;  
  
    // Implicit cast (char to  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an  
    byte bByte = 65;  
    cChar = (char)bByte; // same as cChar = 'A';  
}
```

```
C:\JavaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:43: error: incompatible types: possible lossy conversion from long to int  
        int x = y;  
                           ^  
1 error  
  
C:\JavaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\JavaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:47: error: incompatible types: possible lossy conversion from int to byte  
        byte narrowdByte = 123456;  
                           ^  
1 error  
  
C:\JavaInDepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\JavaInDepth\src\com\semanticsquare\basics>cls
```

Incompatible Type and Information loss due to out-of-range assignment

Example 4:

```
byte narrowdByte = (byte)123456;  
System.out.println("narrowdByte: " + narrowdByte);
```

Since 123456 (Int literal) is out of byte range, with explicit cast will try to store lower/right side bits of binary representation of that decimal numbers.

Hence we will get the result as **64**

Information Loss in Explicit Casting

▶ Out-of-range assignments

- byte narrowedByte = **(byte) 123456; // 64**



▶ Truncation

- Floating-point to integer/char will always truncate
- int x = **(int) 3.14f; // x = 3**
- int y = **(int) 0.9; // y = 0**
- char c = **(char) 65.5; // c = 'A'**



36. Variables: Type Casting + Demo

```
static void typeCasting() {  
    System.out.println("\nIn type casting ...");  
    // Explicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to conversion  
    byte narrowdByte = 127;  
    System.out.println("narrowedByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int)0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to char)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // same as (char)(byte)bByte
```

```
C:\Windows\system32\cmd.exe  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:43: error: incompatible types: possible lossy conversion from long to int  
        int x = y;  
                           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:47: error: incompatible types: possible lossy conversion from int to byte  
        byte narrowdByte = 123456;  
                           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
  
Inside typeCasting ...  
narrowdByte: 64  
iTruncated: 0  
iInt: 65  
cChar: A  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:47: error: incompatible types: possible lossy conversion from int to byte  
        byte narrowdByte = 128;  
                           ^  
1 error  
  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>
```

```
// Truncation  
Cast a floating point to an integer/char variable, will always truncate the number.
```

Example 5:

```
int iTruncated = (int)0.99;  
System.out.println("iTruncated: " + iTruncated);
```

Above code will print
iTruncated: 0

```
// Implicit cast (int to long)  
Example 6:
```

```
Int x = 65  
long y;  
y = x;
```

Above code will run fine and assign y as 65.

```
// Implicit cast (char to int)  
Example 7:  
char cChar = 'A';  
int iInt = cChar;
```

```
System.out.println("iInt: " + iInt);
```

No explicit cost is required, but out of range assignments would also lead to information loss.

```
va
System.out.println("narrowdByte: " + narrowdByte);

// Truncation
int iTruncated = (int)0.99;
System.out.println("iTruncated: " + iTruncated);

// Implicit cast (int to long)
y = x;

// Implicit cast (char to int)
char cChar = 'A';
int iInt = cChar;
System.out.println("iInt: " + iInt);

// byte to char using an explicit cast
byte bByte = 65;
cChar = (char)bByte; // special conversion (widening from byte --> int followed by
                     // narrowing from int --> char)
System.out.println("cChar: " + cChar);
}

public static void main(String[] args) {
    // Language Basics 1
    //print();
    //primitives();
    typeCasting();
}
```

```
// special conversion (widening from byte --> int followed by narrowing from int --> char)
```

Example 7 :

```
byte bByte = 65;  
cChar = (char)bByte; // special conversion  
  
System.out.println("cChar: " + cChar);
```

36. Variables: Type Casting + Demo

```
static void typeCasting() {  
    System.out.println("\nInside typeCasting ...");  
    // Explicit casting  
    long y = 42;  
    //int x = y;  
    int x = (int)y;  
  
    // Information loss due to type conversion  
    byte narrowdByte = (byte) y;  
    System.out.println("narrowedByte: " + narrowdByte);  
  
    // Truncation  
    int iTruncated = (int) 0.9;  
    System.out.println("iTruncated: " + iTruncated);  
  
    // Implicit cast (int to char)  
    y = x;  
  
    // Implicit cast (char to int)  
    char cChar = 'A';  
    int iInt = cChar;  
    System.out.println("iInt: " + iInt);  
  
    // byte to char using an explicit cast  
    byte bByte = 65;  
    cChar = (char)bByte; // same as (char)(byte)bByte;  
    System.out.println("cChar: " + cChar);  
}
```

```
C:\Windows\system32\cmd.exe  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside typeCasting ...  
narrowdByte: 64  
iTruncated: 0  
iInt: 65  
cChar: A  
C:\javaindepth\src\com\semanticsquare\basics>
```


Casting Use-cases

▶ Implicit Casting

```
float f1 = 3.133f;  
float f2 = 4.135f;  
go(f1, f2);
```

```
go(double d1, double d2) {  
    ...  
}
```

▶ Explicit casting

```
double avg = (2+3)/2; // 2.0, not 2.5  
double avg = (double) (2+3)/2;
```

Use cases where casting is applied:

- Let's assume that we have two variables, F1 and F2, Next, we can call the method go by passing the float variables as input, the values will be assigned to the two double parameters that the method defines.
- Next, with regards to explicit casting, let's consider this example where we want to compute average of two numbers.

The result for 2 integer will be integer , so $(2+3)/2$ this will give 2 of integer type instead of 2.5.
After we pass it to double , it will be converted into 2.0 instead of 2.5. Hence we need to and explicit cast here.

Variables: Object References

Dheeru Mundluru

Object References

(reference type)

`Student s = new Student();`

Allocate space for
reference variable

Allocate space for
new student object

`s ← student object's address`

Now, let's look at the second type of variables which can hold **object references**.

This statement creates a student object.

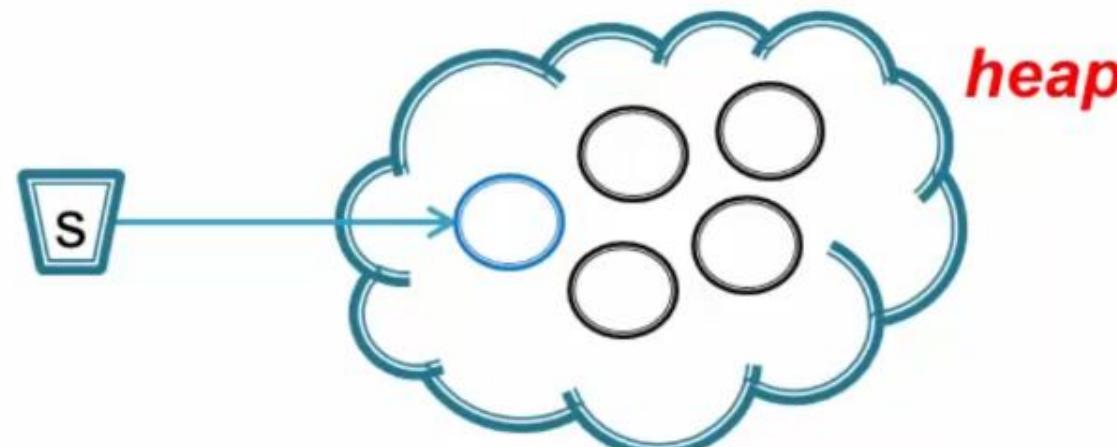
s is an **object reference** here and type is a **reference type(Class / Interface)**.

The statement has 3 parts on it, **each informs JVM to do something**.

1. First part **Student s** allocate memory for reference variable.
2. The second part **new student()** allocate memory for a new student object.
3. Finally, the assignment operator assigns the student object to the reference variable.

Where are Objects Stored?

Objects live on *heap*



When JVM starts up, it gets a chunk of memory from the underlying operating system. one area of this memory, is heap.

All objects are stored on the heap. (Above diagram shows the heap with all the objects).

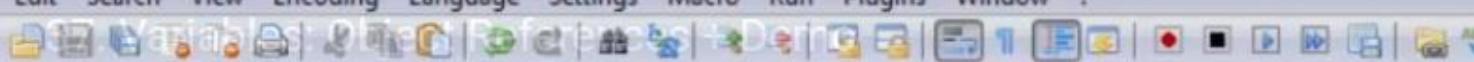
The variable references (example: `s`) will point relevant object on the heap.

Bit Depth & Default

- ▶ Bit depth ~ JVM specific
- ▶ Default ~ **null**
 - Student s;
 - s.updateProfile(); // **NullPointerException**



- All object references have the same big depth (and bit size is JVM specific) regardless of objects they reference to.
- Default value for an object reference is **null**, it implies object reference is not pointing to anything.
- Use of object reference with null value gives run time exception called **null pointer exception**.



Example 1 :

Usage of object dot operator (.) with reference variable gives null pointer exception.

```
Edit Search View Encoding Language Settings Ma
ClientA.java ClientB.java JarHell.java myins new
6
7     student2.name
8     student2.gender
9     student2.age =
10    student2.phone
11    student2.gpa =
12    student2.degree
13    student2.inter
14    student2.compu
15
16    //Student stu
17    student3.id =
18    student3.name
19    student3.gender
20    student3.age =
21    student3.phone
22    student3.gpa =
23    student3.degree
24    student3.inter
25    student3.compu
26
27    System.out.pr
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99 sourcefile
```

C:\javaindepth\src\com\semanticsquare\basics>javac Student.java

C:\javaindepth\src\com\semanticsquare\basics>java Student

id: 1000
nextId: 1001
name: John
gender: male
age: 18
phone: 2234567890
gpa: 3.8
degree: B
tuitionFees: 12000.0
computeCount: 1

id: 1001
nextId: 1002
name: Raj
gender: male
age: 21
phone: 2234569999
gpa: 3.4
degree: M
tuitionFees: 17000.0
computeCount: 2

Exception in thread "main" java.lang.NullPointerException
at Student.main(Student.java:66)

C:\javaindepth\src\com\semanticsquare\basics>

Statements

Dheeru Mundluru

Statement Types

- ▶ Declaration statements, e.g., int **count** = 25;
- ▶ Expression statements
 - count = 25; // *assignment statement*
 - getCount(); // *method invocation statement*
 - count++; // *increment statement*
- ▶ Control flow statements
 - if** (count < 100) {
 ...
}

not at class-level

Statements can be of 3 kinds.

Declaration statements :

They basically declare **variables**, **methods**, **constructors**, **nested class**, **interfaces**. **instance initializer**, **static initializer**.

These are the only statements which can be written at Class level.

Expressions statements :

If expressions appear separately (from declaration) with a semicolon, then they are referred to as expressions statements.

Control statements :

Control statements regulate the order in which statements get executed.

38. Statements + Demo

```
int iTruncated = (int)0.99;
System.out.println("iTruncated: " + iTruncated);

// Implicit cast (int to long)
y = x;

// Implicit cast (char to int)
char cChar = 'A';
int iInt = cChar;
System.out.println("iInt: " + iInt);

// byte to char using an explicit cast
byte bByte = 65;
cChar = (char)bByte; // special conversion (widening from byte --> int followed by
narrowing from int --> char)
System.out.println("cChar: " + cChar);
}

static int count = 25;

public static void main(String[] args) {
    // Language Basics 1
    //print();
    //primitives();
    typeCasting();
}
```

Example 1 :

Declaration statement at class level run fine

Static int count = 25;

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

C:\javaindepth\src\com\semanticsquare\basics>

100

100

100

104

100

100

100

100

104

10

104

100

100

104

100

100

100

104

followed by

38. Statements + Demo

```
int iInt = cChar;
System.out.println("iInt: " + iInt);

// byte to char using an explicit cast
byte bByte = 65;
cChar = (char)bByte; // special conversion (widening from byte --> int followed by
narrowing from int --> char)
System.out.println("cChar: " + cChar);
}

static int count = 25;
count = 34;

public static void main(String[] args) {
    // Language Basics 1
    //print();
    //primitives();
    typeCasting();
}

// Field declarations
// Method declarations
// Constructor declarations
// Nested class & interface declarations
// Instance and static initializers
```

Example 2 :

Expression statement at class level gives compilation error “**Identifier Expected**”

count = 14;

38. Statements + Demo

```
int i;
System.out.println("Count is " + count);
count = 34;
// by
byte b;
char cChar;
narrow();
System.out.println("Count is " + count);

static {
    count = 34;
}

public static void main(String[] args) {
    // Lambda expression
    // print
    // print
    typeCast();
}

// Fields
// Methods
// Constructors
// Nested classes
// Instances
```

followed by

38. Statements + Demo

```
int iInt = cChar;
System.out.println("iInt: " + iInt);

// byte to char using an explicit cast
byte bByte = 65;
cChar = (char)bByte; // special conversion (widening from byte --> int followed by
narrowing from int --> char)
System.out.println("cChar: " + cChar);
}

static int count = 25;

if (count < 49) {
}

public static void main(String[] args) {
    count = 34;
    // Language Basics 1
    //print();
    //primitives();
    typeCasting();
}

// Field declarations
// Method declarations
// Constructor declarations
```

Example 3 :

Control statement at class level gives compilation error “**illegal start of type if**”s

```
if (count < 14)
{
}
```

38. Statements + Demo

```
int i;
System.out.println("Count = " + count);
// by default, println() prints a new line
cChar = 'A';
narrow((char) cChar);
System.out.println("Count = " + count);
}
static void main(String[] args) {
    if (count < 49) {
        System.out.println("Count = " + count);
    }
    public static void main(String[] args) {
        //count = 34;
        // Large number
        //print(count);
        //print("Count = " + count);
        typeCount();
    }

    // Fields
    // Methods
    // Constructors
    // ...
}
```

followed by



Arrays

Dheeru Mundluru

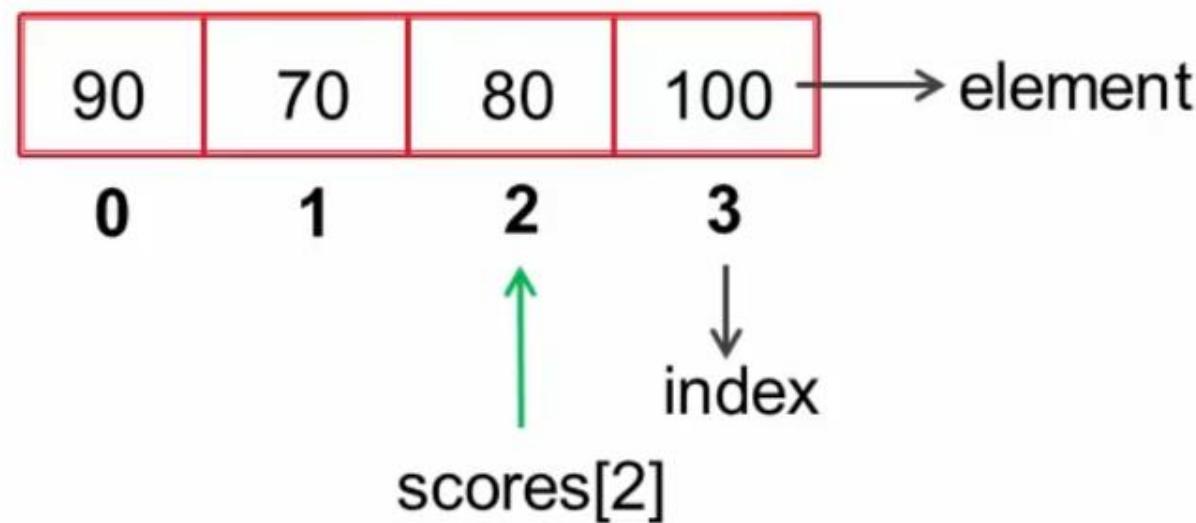


Sometimes we need collection of similar type of values.

Like , if we have to keep track of all the course titles that a student has registered for, then instead of having a separate variable to hold each of those course titles. we can store all those course titles in a single data structure which is represented by some variable. This collection is **Array**.

Arrays

Container **object** that holds **fixed #** values of **single type**



An array is basically

- **An object** (stored on the heap).
- Stores items of **same data type**.
- **Size** (number of items that can store) is specified **at the creation time** and cannot be changed.
- The index starts from **0**.

Example 1: [90][70][80][100]

Declaration option :

- 1. Int[] scores = new int[4];**
- 2. Int scores[] = new int[4];**
- 3. Int[] scores = new int[] {90, 70, 80, 100};**
- 4. Int[] scores = {90, 70, 80, 100};** // This format will not work during reinitialization, you will need new keyword there.

Reinitialization option :

- 1. scores[0] = 90 ;**
- 2.**
static int[] scores[];
scores = new int[4];
scores[0] = 90;
- 3.**
static int[] scores[];
scores = new int[] {90, 70, 80, 100};

Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo

Client.java ClientB.java JarHelp.java my.inr new_7 new_6 Student.java BasicsDemo.java new_8

```
1     System.out.println("iInt: " + iInt);
2
3     // byte to char using an explicit cast
4     byte bByte = 65;
5     cChar = (char)bByte; // special conversion (widening from byte --> int followed by
6     narrowing from int --> char)
7     System.out.println("cChar: " + cChar);
8 }
9
10 static void arrays() {
11     System.out.println("\nInside arrays ...");
12     int[] scores = new int[4];
13     System.out.println("Mid-Term 1: " + scores[0]);
14     System.out.println("Mid-Term 2: " + scores[1]);
15     System.out.println("Final: " + scores[2]);
16     System.out.println("Project: " + scores[3]);
17 }
18
19 public static void main(String[] args) {
20     // Language Basics 1
21     //print();
22     //primitives();
23     //typeCasting();

```

length : 2843 lines : 85 Ln : 75 Col : 38 Sel : 0 Dos\Windows ANSI CC INS

source file 1x 7:21 / 27:35

Array Creation approach 1

Example 1 :

```
Static void array()
{
    Int[] scores = new int[4];

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);

}
```

Since the array is part of static member , all the array items will get default value.
Hence above code will print 0 for all 4 integer items.

39. Arrays + Demo

```
System.out.println("Inside arrays ...");
byte bByte = 6;
cChar = (char) bByte;
System.out.println("Mid-Term 1: " + cChar);
System.out.println("Mid-Term 2: " + cChar);
System.out.println("Final: " + cChar);
System.out.println("Project: " + cChar);
```

Output:

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 0
Mid-Term 2: 0
Final: 0
Project: 0
```


Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo

Client.java ClientB.java JarHello.java my.in1 new_7 new_6 Student.java BasicsDemo.java new_8

```
1     System.out.println("iInt: " + iInt);
2
3     // byte to char using an explicit cast
4     byte bByte = 65;
5     cChar = (char)bByte; // special conversion (widening from byte --> int followed by
6     narrowing from int --> char)
7     System.out.println("cChar: " + cChar);
8 }
9
10 static void arrays() {
11     System.out.println("\nInside arrays ...");
12     int[] scores = new int[4];
13     scores[0] = 90;
14     scores[1] = 70;
15     scores[2] = 80;
16     scores[3] = 100;
17     System.out.println("Mid-Term 1: " + scores[0]);
18     System.out.println("Mid-Term 2: " + scores[1]);
19     System.out.println("Final: " + scores[2]);
20     System.out.println("Project: " + scores[3]);
21 }
22
23 public static void main(String[] args) {
```

source file 1x 8:31 / 27:35 length : 2928 lines : 89 Ln : 75 Col : 26 Sel : 0 Dos\Windows ANSI CC IN

Array Creation approach 1

Example 2 :

```
Static void array()
{
    Int[] scores = new int[4];
    scores[0] = 90 ;
    scores[1] = 70 ;
    scores[2] = 80 ;
    scores[3] = 100 ;
    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);

}
```

above code will print respective value of each integer items.

Edit Search View Encoding Language Settings Help

39. Arrays + Demo

Client.java ClientB.java JarHell.java my.inn new

```
1 System.out.println("Inside arrays ...");
2
3         // byte to char
4         byte bByte = 65;
5         cChar = (char) bByte;
6         narrowing from byte to char
7         System.out.println("Mid-Term 1: " + MidTerm1);
8         System.out.println("Mid-Term 2: " + MidTerm2);
9         System.out.println("Final: " + Final);
10        System.out.println("Project: " + Project);
11
12    }
13
14
15    static void arrays()
16    {
17        System.out.println("Inside arrays ...");
18        int[] scores = {90, 70, 80, 100};
19        scores[0] = 90;
20        scores[1] = 70;
21        scores[2] = 80;
22        scores[3] = 100;
23
24        System.out.println("Mid-Term 1: " + scores[0]);
25        System.out.println("Mid-Term 2: " + scores[1]);
26        System.out.println("Final: " + scores[2]);
27        System.out.println("Project: " + scores[3]);
28    }
29
30
31    public static void main(String[] args)
32    {
33        arrays();
34    }
35
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 0
Mid-Term 2: 0
Final: 0
Project: 0
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100
C:\javaindepth\src\com\semanticsquare\basics>
```


39.Arrays + Demo

```
1 System.out.println("Inside arrays ...");
2     Mid-Term 1: 0
3     Mid-Term 2: 0
4     Final: 0
5     Project: 0
6 }
7
8 static void arrays()
9 {
10    System.out.println("Inside arrays ...");
11    int scores[] = { 90, 70, 80, 100 };
12    scores[0] = 90;
13    scores[1] = 70;
14    scores[2] = 80;
15    scores[3] = 100;
16    System.out.println("Mid-Term 1: " + scores[0]);
17    System.out.println("Mid-Term 2: " + scores[1]);
18    System.out.println("Final: " + scores[2]);
19    System.out.println("Project: " + scores[3]);
20 }
21
22 public static void main(String[] args)
23 {
24 }
```

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ...
Mid-Term 1: 0
Mid-Term 2: 0
Final: 0
Project: 0

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100

C:\javaindepth\src\com\semanticsquare\basics>

Array Creation approach 2

Example 3 :

```
//original way  
//Int[] scores = new int[4];  
  
//alternate way  
Int scores[] = new int[4];
```

We can also write Array declaration as above.

Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo Client.java ClientB.java JarHell.java my.in1 new_7 new_6 Student.java BasicsDemo.java new_8

```
1     System.out.println("iInt: " + iInt);
2
3     // byte to char using an explicit cast
4     byte bByte = 65;
5     cChar = (char)bByte; // special conversion (widening from byte --> int followed by
6     narrowing from int --> char)
7     System.out.println("cChar: " + cChar);
8 }
9
10 static void arrays() {
11     System.out.println("\nInside arrays ...");
12     int[] scores = new int[4];
13     scores[0] = 90;
14     scores[1] = 70L; I
15     scores[2] = 80;
16     scores[3] = 100;
17     System.out.println("Mid-Term 1: " + scores[0]);
18     System.out.println("Mid-Term 2: " + scores[1]);
19     System.out.println("Final: " + scores[2]);
20     System.out.println("Project: " + scores[3]);
21 }
22
23 public static void main(String[] args) {
```

length : 2929 lines : 89 Ln : 73 Col : 26 Sel : 0 Dos\Windows ANSI CC INS

source file 1x 9:18 / 27:35

Array Creation approach 1

Example 4 :

```
Static void array()
{
    Int[] scores = new int[4];
    scores[0] = 90 ;
    scores[1] = 70L ;
    scores[2] = 80 ;
    scores[3] = 100 ;
    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);
}
```

above code will give compilation error “Incompatible types” , because we are trying to put long literal in int variable without any explicit type casting.

Project: 100

39. Arrays + Demo

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:73: error: incompatible types: possible lossy conversion from long to int
        scores[1] = 70L;
                           ^
1 error
```

C:\javaindepth\src\com\semanticsquare\basics>

int followed by

source file 9:36 / 27:35

os\Windows ANSI CC INS

© semanticsquare.com



ClientA.java ClientB.java JarHell.java my.inr new_7 new_6 Student.java BasicsDemo.java new_8

```
1     System.out.println("iInt: " + iInt);

2     // byte to char using an explicit cast
3     byte bByte = 65;
4     cChar = (char)bByte; // special conversion (widening from byte --> int followed by
5     narrowing from int --> char)
6     System.out.println("cChar: " + cChar);
7 }

8 static void arrays() {
9     System.out.println("\nInside arrays ...");
0     /*int[] scores = new int[4];
1     scores[0] = 90;
2     scores[1] = 70;
3     scores[2] = 80;
4     scores[3] = 100;*/
5     int[] scores = new int[] {90, 70, 80, 100};
6     System.out.println("Mid-Term 1: " + scores[0]);
7     System.out.println("Mid-Term 2: " + scores[1]);
8     System.out.println("Final: " + scores[2]);
9     System.out.println("Project: " + scores[3]);
0 }

1
2
```

Array Creation approach 3

Example 5 :

```
Static void array()
{
    Int[] scores = new int[] {90, 70, 80, 100};

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);

}
```

above code will run fine

Edit Search View Encoding Language Settings Macro Run

39. Arrays + Demo

```
1     System.out.println  
2  
3         // byte to char us  
4         byte bByte = 65;  
5         cChar = (char)bByt  
6         narrowing from int  
7         System.out.println  
8     }  
9  
10    static void arrays() {  
11        System.out.println  
12        /*int[] scores = n  
13        scores[0] = 90;  
14        scores[1] = 70;  
15        scores[2] = 80;  
16        scores[3] = 100;*/  
17        int[] scores = new  
18        System.out.println  
19        System.out.println  
20        System.out.println  
21        System.out.println  
22    }
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside arrays ...  
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
C:\javaindepth\src\com\semanticsquare\basics>
```


Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo Client.java ClientB.java JarHello.java my.in1 new_7 new_6 Student.java BasicsDemo.java new_8

```
1     System.out.println("iInt: " + iInt);
2
3     // byte to char using an explicit cast
4     byte bByte = 65;
5     cChar = (char)bByte; // special conversion (widening from byte --> int followed by
6     narrowing from int --> char)
7     System.out.println("cChar: " + cChar);
8 }
9
10 static void arrays() {
11     System.out.println("\nInside arrays ...");
12     /*int[] scores = new int[4];
13     scores[0] = 90;
14     scores[1] = 70;
15     scores[2] = 80;
16     scores[3] = 100;*/
17     int[] scores = new int[4] {90, 70, 80, 100};
18     System.out.println("Mid-Term 1: " + scores[0]);
19     System.out.println("Mid-Term 2: " + scores[1]);
20     System.out.println("Final: " + scores[2]);
21     System.out.println("Project: " + scores[3]);
22 }
```

source file 1x 11:17 / 27:35 length : 2986 lines : 90 Ln : 76 Col : 35 Sel : 0 Dos\Windows ANSI CC IN

Array Creation approach 3

Example 6 :

```
Static void array()
{
    Int[] scores = new int[4] {90, 70, 80, 100};

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);

}
```

above code will give compilation error “**Array creation with both , dimension expression and initialization is illegal**”.

Edit Search View Encoding Language Settings Macro Run

39. Arrays + Demo

```
1     System.out.println
2
3         // byte to char us
4         byte bByte = 65;
5         cChar = (char)bByte;
6         narrowing from int
7         System.out.println
8
9     }
10
11     static void arrays() {
12         System.out.println
13         /*int[] scores = new int[4];
14         scores[0] = 90;
15         scores[1] = 70;
16         scores[2] = 80;
17         scores[3] = 100; */
18         int[] scores = new int[4];
19         scores[0] = 90;
20         scores[1] = 70;
21         scores[2] = 80;
22         scores[3] = 100;
23
24         System.out.println
25         System.out.println
26         System.out.println
27         System.out.println
28     }
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:76: error: array creation with both dimension expression and initialization is illegal
        int[] scores = new int[4] {90, 70, 80, 100};
                                         ^
1 error

C:\javaindepth\src\com\semanticsquare\basics>
```



```
7
8
9     static void arrays() {
10        System.out.println("\nInside arrays ...");
11        /*int[] scores = new int[4];
12        scores[0] = 90;
13        scores[1] = 70;
14        scores[2] = 80;
15        scores[3] = 100;*/
16
17        //int[] scores = new int[] {90, 70, 80, 100};
18
19        int[] scores = {90, 70, 80, 100};
20
21        System.out.println("Mid-Term 1: " + scores[0]);
22        System.out.println("Mid-Term 2: " + scores[1]);
23        System.out.println("Final: " + scores[2]);
24        System.out.println("Project: " + scores[3]);
25    }
26
27    public static void main(String[] args) {
28        // Language Basics 1
29        //print();
30    }
31
32}
```

Array Creation approach 3

Example 5 :

```
// Previous approach  
//Int[] scores = new int[] {90, 70, 80, 100};
```

```
// New approach  
Int[] scores = {90, 70, 80, 100};
```

above code will run fine by declaring the array with new approach too.



ClientA.java ClientB.java JarHell.java my.in new_7 new_8

```
7 }  
8  
9     static void arrays() {  
0         System.out.println  
1             /*int[] scores = n  
2             scores[0] = 90;  
3             scores[1] = 70;  
4             scores[2] = 80;  
5             scores[3] = 100;*/  
6  
7             //int[] scores = n  
8                 I  
9                 int[] scores = {90}  
0  
1                 System.out.println  
2                 System.out.println  
3                 System.out.println  
4                 System.out.println  
5 }  
6  
7     public static void main  
8         // Language Basics  
9         //print();
```

source file

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside arrays ...  
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
BasicsDemo.java:76: error: array creation with both dimension expression and ini  
tialization is illegal  
                                int[] scores = new int[4] {90, 70, 80, 100};  
                                         ^  
1 error  
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside arrays ...  
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
C:\javaindepth\src\com\semanticsquare\basics>
```



```
7
8
9     static int[] scores;
0
1     static void arrays() {
2         System.out.println("\nInside arrays ...");
3         /*int[] scores = new int[4];
4         scores[0] = 90;
5         scores[1] = 70;
6         scores[2] = 80;
7         scores[3] = 100;*/
8
9         //int[] scores = new int[] {90, 70, 80, 100};
0
1         //int[] scores = {90, 70, 80, 100};
2
3         System.out.println("Mid-Term 1: " + scores[0]);
4         System.out.println("Mid-Term 2: " + scores[1]);
5         System.out.println("Final: " + scores[2]);
6         System.out.println("Project: " + scores[3]);
7
8
9     public static void main(String[] args) {
```

Array Creation approach 4

Example 6 :

```
static int[] scores;  
  
Static void array()  
{  
  
    System.out.println("Mid-Term 1 : " + scores[0]);  
    System.out.println("Mid-Term 2 : " + scores[1]);  
    System.out.println("Final : " + scores[2]);  
    System.out.println("Project : " + scores[3]);  
  
}
```

above code will give compilation error “null pointer exception” , because array is neither initialized using size nor items, so the size of array is unknown to allocate array’s memory.

Edit Search View Encoding Language Settings Macro Run

39. Arrays + Demo

```
9 static int[] scores;  
10  
11 static void arrays() {  
12     System.out.println(  
13         /*int[] scores = n  
14         scores[0] = 90;  
15         scores[1] = 70;  
16         scores[2] = 80;  
17         scores[3] = 100;*/  
18  
19 //int[] scores = n  
20  
21 //int[] scores = {  
22  
23     System.out.println(  
24     System.out.println(  
25     System.out.println(  
26     System.out.println(  
27 }  
28  
29 public static void mai  
30     // Language Basics  
31     //print();
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside arrays ...  
Exception in thread "main" java.lang.NullPointerException  
    at BasicsDemo.arrays(BasicsDemo.java:83)  
    at BasicsDemo.main(BasicsDemo.java:94)
```

C:\javaindepth\src\com\semanticsquare\basics>



```
9 static int[] scores;  
10  
11 static void arrays() {  
12     System.out.println("\nInside arrays ...");  
13     //int[] scores = new int[4];  
14     scores = new int[4];  
15     scores[0] = 90;  
16     scores[1] = 70;  
17     scores[2] = 80;  
18     scores[3] = 100;  
19  
20     //int[] scores = new int[] {90, 70, 80, 100};  
21  
22     //int[] scores = {90, 70, 80, 100};  
23  
24     System.out.println("Mid-Term 1: " + scores[0]);  
25     System.out.println("Mid-Term 2: " + scores[1]);  
26     System.out.println("Final: " + scores[2]);  
27     System.out.println("Project: " + scores[3]);  
28 }  
29  
30 public static void main(String[] args) {  
31     // Language Basics 1
```

Array Creation approach 5

Example 6 :

```
static int[] scores[];  
  
Static void array()  
{  
    scores = new int[4];  
    scores[0] = 90;  
    scores[1] = 70;  
    scores[2] = 80;  
    scores[3] = 100;  
  
    System.out.println("Mid-Term 1 : " + scores[0]);  
    System.out.println("Mid-Term 2 : " + scores[1]);  
    System.out.println("Final : " + scores[2]);  
    System.out.println("Project : " + scores[3]);  
}
```

above code will work fine.

Edit Search View Encoding Language Settings Macro Run

39.Arrays + Demo Client.java ClientB.java JarHell.java my.inn new_7 new_8

```
static int[] scores;

static void arrays() {
    System.out.println
    //int[] scores = n
    scores = new int[4]
    scores[0] = 90;
    scores[1] = 70;
    scores[2] = 80;
    scores[3] = 100;

    //int[] scores = n

    //int[] scores = {

        System.out.println
        System.out.println
        System.out.println
        System.out.println
    }

    public static void mai
        // Language Basics
}
```

Command Prompt

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100
C:\javaindepth\src\com\semanticsquare\basics>
```

source file 1x 14:22 / 27:35 © semanticsquare.com

Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo Client.java ClientB.java JarHello.java my.in1 new_7 new_6 Student.java BasicsDemo.java new_8

```
9 static int[] scores;
0
1 static void arrays() {
2     System.out.println("\nInside arrays ...");
3     /*int[] scores = new int[4];
4
5     scores[0] = 90;
6     scores[1] = 70;
7     scores[2] = 80;
8     scores[3] = 100;*/
9
0     /*int[] scores = new int[] {90, 70, 80, 100};*/
1
2     /*int[]*/ scores = {90, 70, 80, 100}; I
3
4     System.out.println("Mid-Term 1: " + scores[0]);
5     System.out.println("Mid-Term 2: " + scores[1]);
6     System.out.println("Final: " + scores[2]);
7     System.out.println("Project: " + scores[3]);
8
9 }
0
1 public static void main(String[] args) {
    // Language Basics 1
```

length : 3079 lines : 97 Ln : 82 Col : 20 Sel : 0 Dos\Windows ANSI CC INS

source file 1x 15:13 / 27:35

Array Creation approach 6

Example 6 :

```
static int[] scores[];
```

```
Static void array()
{
    scores = {90, 70, 80, 100};
```

```
System.out.println("Mid-Term 1 : " + scores[0]);
System.out.println("Mid-Term 2 : " + scores[1]);
System.out.println("Final : " + scores[2]);
System.out.println("Project : " + scores[3]);
```

```
}
```

above code give run time error, "**illegal start of expression**"

Because reinitialization using {90, 70, 80, 100} type of syntax directly is not possible, we will need new keyword to use such format.

39 Arrays + Demo

```
9 static int[] scores;
0
1 static void arrays() {
2     System.out.println(
3         /*int[] scores = n
4
5         scores[0] = 90;
6         scores[1] = 70;
7         scores[2] = 80;
8         scores[3] = 100; */
9
0     /*int[] scores = n
1
2     /*int[]*/ scores =
3
4     System.out.println(
5         System.out.println(
6             System.out.println(
7                 System.out.println(
8                     }
9
0     public static void mai
1         // Language Basics
```

```
ON Command Prompt

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:82: error: illegal start of expression
        /*int[]*/ scores = {90, 70, 80, 100};
                           ^
BasicsDemo.java:82: error: not a statement
        /*int[]*/ scores = {90, 70, 80, 100};
                           ^
BasicsDemo.java:82: error: ';' expected
        /*int[]*/ scores = {90, 70, 80, 100};
                           ^
3 errors

C:\javaindepth\src\com\semanticsquare\basics>
```



```
9 static int[] scores;  
10  
11 static void arrays() {  
12     System.out.println("\nInside arrays ...");  
13     /*int[] scores = new int[4];  
14     |  
15     scores[0] = 90;  
16     scores[1] = 70;  
17     scores[2] = 80;  
18     scores[3] = 100; */  
19  
20     /*int[] */ scores = new int[] {90, 70, 80, 100};  
21  
22     //int[] scores = {90, 70, 80, 100};  
23  
24     System.out.println("Mid-Term 1: " + scores[0]);  
25     System.out.println("Mid-Term 2: " + scores[1]);  
26     System.out.println("Final: " + scores[2]);  
27     System.out.println("Project: " + scores[3]);  
28 }  
29  
30 public static void main(String[] args) {  
31     // Language Basics 1
```

Array Creation approach 6

Example 6 :

```
static int[] scores[];  
  
Static void array()  
{  
    scores = new int[] {90, 70, 80, 100};  
  
    System.out.println("Mid-Term 1 : " + scores[0]);  
    System.out.println("Mid-Term 2 : " + scores[1]);  
    System.out.println("Final : " + scores[2]);  
    System.out.println("Project : " + scores[3]);  
}
```

above code will work fine.

20 Arrows & Doms

```
9 static int[] scores;  
0  
1 static void arrays() {  
2     System.out.println(  
3         /*int[] scores = n  
4             *  
5             scores[0] = 90;  
6             scores[1] = 70;  
7             scores[2] = 80;  
8             scores[3] = 100; */  
9  
0         /*int[] */ scores  
1  
2         //int[] scores = {  
3             System.out.println(  
4                 System.out.println(  
5                     System.out.println(  
6                         System.out.println(  
7                             System.out.println(  
8                         })  
9  
0         public static void main  
1             // Language Basics
```

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ..

Mid-Term 1: 90
Mid-Term 2: 70

Final: 80
Project: 100

C:\javainde

C:\javaindent\src\com\semanticsquare\basics>java BasicsDemo

C:\javaindepth\src\com\semanticsquare\basics> java BasicsDemo

Inside arrays ..

Mid-Term 1: 90
Mid-Term 2: 70
Final: 80

Final: 80
Project: 100

E:\javaindepth\src\com\semanticsquare\basics



ClientA.java

ClientB.java

JarHell.java

my.ini

new_7

new_6

Student.java

BasicsDemo.java

new_8

```
9  
0 static void arrays() {  
1     System.out.println("\nInside arrays ...");  
2     int[] scores = new int[4];  
3  
4     scores[0] = 90;  
5     scores[1] = 70;  
6     scores[2] = 80;  
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1     //int[] scores = {90, 70, 80, 100};  
2  
3     System.out.println("Mid-Term 1: " + scores[0]);  
4     System.out.println("Mid-Term 2: " + scores[1]);  
5     System.out.println("Final: " + scores[2]);  
6     System.out.println("Project: " + scores[3]);  
7     System.out.println("# exams: " + scores.length);  
8  
9 }  
0  
1 public static void main(String[] args) {  
2     // Language Basics 1
```

Array Length

Example 1 :

```
Static void array()
{
    int[] scores = new int[4];

    scores[0] = 90;
    scores[1] = 70;
    scores[2] = 80;
    scores[3] = 100;

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[3]);
    System.out.println("Number of items/exams : " + scores.length );

}
```

above code run fine and give the number of items in the array.

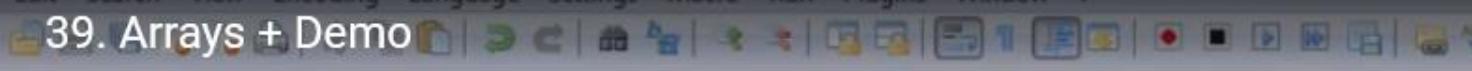
```
C:\javaindepth\src\com\semanticsquare\basics> javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics> java BasicsDemo
Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100
# exams: 4
C:\javaindepth\src\com\semanticsquare\basics>
```



```
9  
0 static void arrays() {  
1     System.out.println("\nInside arrays ...");  
2     int[] scores = new int[4];  
3  
4     scores[0] = 90;  
5     scores[1] = 70;  
6     scores[2] = 80;  
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1     //int[] scores = {90, 70, 80, 100};  
2  
3     System.out.println("Mid-Term 1: " + scores[0]);  
4     System.out.println("Mid-Term 2: " + scores[1]);  
5     System.out.println("Final: " + scores[2]);  
6     System.out.println("Project: " + scores[3]);  
7     System.out.println("# exams: " + scores.length());  
8  
9 }  
0  
1 public static void main(String[] args) {  
2     // Language Basics 1
```

Length is an Attribute of Array object not method.

Hence **scores.length** will work fine but not **scores.length()**



```
9  
0     static void arrays() {  
1         System.out.println("\nInside arrays ...");  
2         int[] scores = new int[4];  
3  
4         scores[0] = 90;  
5         scores[1] = 70;  
6         scores[2] = 80;  
7         scores[3] = 100;  
8  
9         /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1         //int[] scores = {90, 70, 80, 100};  
2  
3         System.out.println("Mid-Term 1: " + scores[0]);  
4         System.out.println("Mid-Term 2: " + scores[1]);  
5         System.out.println("Final: " + scores[2]);  
6         System.out.println("Project: " + scores[-1]);  
7         System.out.println("# exams: " + scores.length);  
8  
9     }  
0  
1     public static void main(String[] args) {  
2         // Language Basics 1
```

Access Array items outside the index range.

Example 1 :

```
Static void array()
{
    int[] scores = new int[4];

    scores[0] = 90;
    scores[1] = 70;
    scores[2] = 80;
    scores[3] = 100;

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[-1]);
    System.out.println("Number of items/exams : " + scores.length );

}
```

above code give run time exception “Array Index Out of bound exception”

```
9
0 static void arrays() {
1     System.out.println
2     int[] scores = new
3
4         scores[0] = 90;
5         scores[1] = 70;
6         scores[2] = 80;
7         scores[3] = 100;
8
9     /*int[] scores = n
0
1 //int[] scores = {
2
3     System.out.println
4     System.out.println
5     System.out.println
6     System.out.println
7     System.out.println
8
9 }
0
1 public static void mai
2 // Language Basics
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Project: 100
# exams: 4

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ...
Mid-Term 1: 90
Mid-Term 2: 70
Final: 80
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index -1 ou
t of bounds for length 4
        at BasicsDemo.arrays(BasicsDemo.java:86)
        at BasicsDemo.main(BasicsDemo.java:95)

C:\javaindepth\src\com\semanticsquare\basics>
```



```
9  
0 static void arrays() {  
1     System.out.println("\nInside arrays ...");  
2     int[] scores = new int[4];  
3  
4     scores[0] = 90;  
5     scores[1] = 70;  
6     scores[2] = 80;  
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1     //int[] scores = {90, 70, 80, 100};  
2  
3     System.out.println("Mid-Term 1: " + scores[0]);  
4     System.out.println("Mid-Term 2: " + scores[1]);  
5     System.out.println("Final: " + scores[2]);  
6     System.out.println("Project: " + scores[4]);  
7     System.out.println("# exams: " + scores.length);  
8  
9 }  
0  
1 public static void main(String[] args) {  
2     // Language Basics 1
```

Access Array items outside the index range.

Example 2 :

```
Static void array()
{
    int[] scores = new int[4];

    scores[0] = 90;
    scores[1] = 70;
    scores[2] = 80;
    scores[3] = 100;

    System.out.println("Mid-Term 1 : " + scores[0]);
    System.out.println("Mid-Term 2 : " + scores[1]);
    System.out.println("Final : " + scores[2]);
    System.out.println("Project : " + scores[4]);
    System.out.println("Number of items/exams : " + scores.length );

}
```

above code give run time exception “Array Index Out of bound exception”

39. Arrays + Demo

```
static void arrays() {
    System.out.println("The scores are:");
    int[] scores = new int[4];
    scores[0] = 90;
    scores[1] = 70;
    scores[2] = 80;
    scores[3] = 100;
    /*int[] scores = new int[4];
    //int[] scores = new int[4];
    System.out.println(scores[0]);
    System.out.println(scores[1]);
    System.out.println(scores[2]);
    System.out.println(scores[3]);
}
public static void main(String[] args) {
    // Language Basics
```

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ..

Mid-Term 1: 90

Mid-Term 2
Final - 80

Final: 80
Project: 100

Project: 1
exams: 6

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ..

Mid-Term 1: 90

Mid-Term 2

```
Final: 80
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 4
```

```
at BasicsDemo.arrays(BasicsDemo.java:86)
at BasicsDemo.main(BasicsDemo.java:95)
```

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo

Inside arrays ..

Mid-Term 1: 90

Mid-Term 2
Final: 90

Final: 80
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4

```
at BasicsDemo.arrays(BasicsDemo.java:86)
at BasicsDemo.main(BasicsDemo.java:95)
```



```
5     scores[1] = 70;
6     scores[2] = 80;
7     scores[3] = 100;

8     /*int[] scores = new int[] {90, 70, 80, 100};*/
9
10    //int[] scores = {90, 70, 80, 100};

11    System.out.println("Mid-Term 1: " + scores[0]);
12    System.out.println("Mid-Term 2: " + scores[1]);
13    System.out.println("Final: " + scores[2]);
14    System.out.println("Project: " + scores[3]);
15    System.out.println("# exams: " + scores.length);

16    Student[] students = new Student[3];
17    System.out.println("Student 1: " + students[0]);
18    System.out.println("Student 2: " + students[1]);
19    System.out.println("Student 3: " + students[2]);
20 }

21 public static void main(String[] args) {
22     // Language Basics 1
23     //print();
24 }
```

Let's see the same creation approach for Array of Objects instead of Array of Int.

Approach 1 :

Example 1 :

```
Student[] student = new Student[3];
```

```
System.out.println("Student 1 : " + student[0]);  
System.out.println("Student 2 : " + student[1]);  
System.out.println("Student 3 : " + student[2]);
```

The above code will run fine and provide the default value of each Array item (null for object).


```
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1  
2     //int[] scores = {90, 70, 80, 100};  
3  
4     System.out.println("Mid-Term 1: " + scores[0]);  
5     System.out.println("Mid-Term 2: " + scores[1]);  
6     System.out.println("Final: " + scores[2]);  
7     System.out.println("Project: " + scores[3]);  
8     System.out.println("# exams: " + scores.length);  
9  
0  
1     Student[] students = new Student[3];  
2     students[0] = new Student();  
3     students[1] = new Student();  
4     students[2] = new Student();  
5     System.out.println("Student 1: " + students[0]);  
6     System.out.println("Student 2: " + students[1]);  
7     System.out.println("Student 3: " + students[2]);  
8 }  
9
```

8 people have written a note here.

```
public static void main(String[] args) {  
    // Language Basics 1
```

Let's see the same creation approach for Array of Objects instead of Array of Int.

Approach 1 :

Example 2 :

```
Student[] students = new Student[3];
```

```
students[0] = new Student();
```

```
students[1] = new Student();
```

```
students[2] = new Student();
```

```
System.out.println("Student 1 : " + students[0]);
```

```
System.out.println("Student 2 : " + students[1]);
```

```
System.out.println("Student 3 : " + students[2]);
```

The above code will run fine and provide the some random value of each Array item.

```
7     scores[3] = 100;  
8  
9     /*int[] scores = n  
10    //int[] scores = {  
11        System.out.println("Mid-Term 1: " + scores[0]);  
12        System.out.println("Mid-Term 2: " + scores[1]);  
13        System.out.println("Final: " + scores[2]);  
14        System.out.println("Project: " + scores[3]);  
15        System.out.println("# exams: " + scores.length);  
16        System.out.println("Student 1: null");  
17        System.out.println("Student 2: null");  
18        System.out.println("Student 3: null");  
19  
20        Student[] students = new Student[3];  
21        students[0] = new Student();  
22        students[1] = new Student();  
23        students[2] = new Student();  
24        System.out.println("Student 1: " + students[0]);  
25        System.out.println("Student 2: " + students[1]);  
26        System.out.println("Student 3: " + students[2]);  
27    }  
28  
29    public static void main(String[] args) {  
30        // Language Basics
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90
```

```
Mid-Term 2: 70
```

```
Final: 80
```

```
Project: 100
```

```
# exams: 4
```

```
Student 1: null
```

```
Student 2: null
```

```
Student 3: null
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90
```

```
Mid-Term 2: 70
```

```
Final: 80
```

```
Project: 100
```

```
# exams: 4
```

```
Student 1: Student@681a9515
```

```
Student 2: Student@3af49f1c
```

```
Student 3: Student@19469ea2
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

8 people have written a note here.

Edit Search View Encoding Language Settings Macro Run Plugins Window ?

39.Arrays + Demo Client.java JarHello.java my.inl new_7 new_6 Student.java BasicsDemo.java new_8

```
7     scores[3] = 100;
8
9     /*int[] scores = new int[] {90, 70, 80, 100};*/
10
11    //int[] scores = {90, 70, 80, 100};
12
13    System.out.println("Mid-Term 1: " + scores[0]);
14    System.out.println("Mid-Term 2: " + scores[1]);
15    System.out.println("Final: " + scores[2]);
16    System.out.println("Project: " + scores[3]);
17    System.out.println("# exams: " + scores.length);
18
19    Student[] students = new Student[] {new Student(), new Student(), new Student()};
20    /*students[0] = new Student();
21     students[1] = new Student();
22     students[2] = new Student();*/
23    System.out.println("Student 1: " + students[0]);
24    System.out.println("Student 2: " + students[1]);
25    System.out.println("Student 3: " + students[2]);
26
27
28
29    public static void main(String[] args) {
30        // Language Basics 1
31    }
32
33
34
35
36
37
38
39
```

length : 3463 lines : 105 Ln : 89 Col : 92 Sel : 0 Dos\Windows ANSI CC INS

source file 1x 23:20 / 27:35

Approach 2 :

Example 1 :

```
Student[] students = new Student[] {new Student(), new Student(), new Student()};
```

```
System.out.println("Student 1 : " + students[0]);
System.out.println("Student 2 : " + students[1]);
System.out.println("Student 3 : " + students[2]);
```

The above code will run fine and provide the default value of each Array item (null for object).

```
7     scores[3] = 100;  
8  
9     /*int[] scores = n  
10    //int[] scores = {  
11        System.out.println("Mid-Term 1: " + scores[0]);  
12        System.out.println("Mid-Term 2: " + scores[1]);  
13        System.out.println("Final: " + scores[2]);  
14        System.out.println("Project: " + scores[3]);  
15        # exams: 4  
16        Student 1: Student@681a9515  
17        Student 2: Student@3af49f1c  
18        Student 3: Student@19469ea2  
19  
20        Student[] students  
21        /*students[0] = new  
22        students[1] = new  
23        students[2] = new  
24        System.out.println("Student 1: " + students[0]);  
25        System.out.println("Student 2: " + students[1]);  
26        System.out.println("Student 3: " + students[2]);  
27    }  
28  
29    public static void main(String[] args) {  
30        // Language Basics
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java  
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo  
Inside arrays ...  
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
# exams: 4  
Student 1: Student@681a9515  
Student 2: Student@3af49f1c  
Student 3: Student@19469ea2  
C:\javaindepth\src\com\semanticsquare\basics>
```

10 people have written a note here.


```
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1     //int[] scores = {90, 70, 80, 100};  
2  
3     System.out.println("Mid-Term 1: " + scores[0]);  
4     System.out.println("Mid-Term 2: " + scores[1]);  
5     System.out.println("Final: " + scores[2]);  
6     System.out.println("Project: " + scores[3]);  
7     System.out.println("# exams: " + scores.length);  
8  
9     Student[] students = {new Student(), new Student(), new Student()};  
0     /*students[0] = new Student();  
1     students[1] = new Student();  
2     students[2] = new Student();*/  
3     System.out.println("Student 1: " + students[0]);  
4     System.out.println("Student 2: " + students[1]);  
5     System.out.println("Student 3: " + students[2]);  
6 }  
7  
8 public static void main(String[] args) {  
9     // Language Basics 1  
10    System.out.println("10 people have written a note here.");  
11 }
```

10 people have written a note here.

Approach 2 :

Example 2 :

```
//Old way  
// Student[] students = new Student[] {new Student(), new Student(), new Student()};  
  
//New way  
Student[] students = {new Student(), new Student(), new Student()};  
  
System.out.println("Student 1 : " + students[0]);  
System.out.println("Student 2 : " + students[1]);  
System.out.println("Student 3 : " + students[2]);
```

The above code will run fine and provide the default value of each Array item (null for object).

```
7     scores[3] = 100;  
8  
9     /*int[] scores = n  
10    //int[] scores = {  
11        System.out.println("Mid-Term 1: " + scores[0]);  
12        System.out.println("Mid-Term 2: " + scores[1]);  
13        System.out.println("Final: " + scores[2]);  
14        System.out.println("Project: " + scores[3]);  
15        System.out.println(" # exams: " + scores.length);  
16  
17        Student[] students = new Student[4];  
18        /*students[0] = new Student("John", 90);  
19        students[1] = new Student("Jane", 70);  
20        students[2] = new Student("Mike", 80);  
21        System.out.println("Student 1: " + students[0]);  
22        System.out.println("Student 2: " + students[1]);  
23        System.out.println("Student 3: " + students[2]);  
24    }  
25  
26    public static void main(String[] args) {  
27        // Language Basics
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
# exams: 4  
Student 1: Student@681a9515  
Student 2: Student@3af49f1c  
Student 3: Student@19469ea2
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90  
Mid-Term 2: 70  
Final: 80  
Project: 100  
# exams: 4  
Student 1: Student@681a9515  
Student 2: Student@3af49f1c  
Student 3: Student@19469ea2
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

10 people have written a note here.



```
7     scores[3] = 100;  
8  
9     /*int[] scores = new int[] {90, 70, 80, 100};*/  
0  
1  
2  
3     System.out.println("Mid-Term 1: " + scores[0]);  
4     System.out.println("Mid-Term 2: " + scores[1]);  
5     System.out.println("Final: " + scores[2]);  
6     System.out.println("Project: " + scores[3]);  
7     System.out.println("# exams: " + scores.length);  
8  
9     Student[] students = {new Student(), new Student(), new Student()};  
0     /*students[0] = new Student();  
1     students[1] = new Student();  
2     students[2] = new Student();*/  
3     students[0].name = "John";  
4     students[1].name = "Raj";  
5     students[2].name = "Anjita";  
6     System.out.println("Student 1: " + students[0]);  
7     System.out.println("Student 2: " + students[1]);  
8     System.out.println("Student 3: " + students[2]);  
9 }
```

10 people have written a note here.

Example 3 :

```
Static void array()
{
    Student[] students = {new Student(), new Student(), new Student()};

    students[0].name = "John";
    students[0].name = "Raj";
    students[0].name = "Anita";

    System.out.println("Student 1 : " + students[0]);
    System.out.println("Student 2 : " + students[1]);
    System.out.println("Student 3 : " + students[2]);
}
```

above code will run fine and return the value under student.name.

```
7     scores[3] = 100;  
8  
9     /*int[] scores = n  
10    //int[] scores = {  
11        System.out.println("Mid-Term 1: " + scores[0]);  
12        System.out.println("Mid-Term 2: " + scores[1]);  
13        System.out.println("Final: " + scores[2]);  
14        System.out.println("Project: " + scores[3]);  
15        System.out.println("# exams: " + scores.length);  
16        System.out.println("Student 1: John");  
17        System.out.println("Student 2: Raj");  
18        System.out.println("Student 3: Anita");  
19  
20        Student[] students = new Student[3];  
21        /*students[0] = new Student("John", 90);  
22        students[1] = new Student("Raj", 70);  
23        students[2] = new Student("Anita", 80);  
24        students[0].name = "John";  
25        students[1].name = "Raj";  
26        students[2].name = "Anita";  
27        System.out.println(students[0].name);  
28        System.out.println(students[1].name);  
29        System.out.println(students[2].name);  
30    }  
31 }
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90
```

```
Mid-Term 2: 70
```

```
Final: 80
```

```
Project: 100
```

```
# exams: 4
```

```
Student 1: John
```

```
Student 2: Raj
```

```
Student 3: Anita
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```

10 people have written a note here.



```
ClientA.java ClientB.java JarHell.java my.ini new_7 new_6 Student.java BasicsDemo.java new_8
7
8         scores[3] = 100;
9
10        /*int[] scores = new int[] {90, 70, 80, 100};*/
11
12        //int[] scores = {90, 70, 80, 100};
13
14        System.out.println("Mid-Term 1: " + scores[0]);
15        System.out.println("Mid-Term 2: " + scores[1]);
16        System.out.println("Final: " + scores[2]);
17        System.out.println("Project: " + scores[3]);
18        System.out.println("# exams: " + scores.length);
19
20        Student[] students = new Student[3];//{new Student(), new Student(), new Student()};
21        students[0] = new Student();
22        students[1] = new Student();
23        // students[2] = new Student();
24        /*students[0].name = "John";
25        students[1].name = "Raj";
26        students[2].name = "Anita";*/
27        System.out.println("Student 1: " + students[0]);
28        System.out.println("Student 2: " + students[1]);
29        System.out.println("Student 3: " + students[2]);
30    }
```

Example 3 :

```
Static void array()
{
    Student[] students = new Student[3];
    students[0] = new Student();
    students[1] = new Student();

    System.out.println("Student 1 : " + students[0]);
    System.out.println("Student 2 : " + students[1]);
    System.out.println("Student 3 : " + students[2]);
}
```

above code will run fine and return random the value for student[0], student[1] and null for student[2].

Edit Search View Encoding Language Settings Macro Run

39. Arrays + Demo

```
7     scores[3] = 100;  
8  
9     /*int[] scores = n  
10    //int[] scores = {  
11        System.out.println("Inside arrays ...");  
12        System.out.println("Mid-Term 1: " + scores[0]);  
13        System.out.println("Mid-Term 2: " + scores[1]);  
14        System.out.println("Final: " + scores[2]);  
15        System.out.println("Project: " + scores[3]);  
16        System.out.println("# exams: " + scores.length);  
17        System.out.println();  
18  
19        Student[] students;  
20        students[0] = new Student();  
21        students[1] = new Student();  
22        // students[2] = new Student();  
23        /*students[0].name = "John";  
24        students[1].name = "Jane";  
25        students[2].name = "Mike";  
26        System.out.println("Student 1: " + students[0].name);  
27        System.out.println("Student 2: " + students[1].name);  
28        System.out.println("Student 3: " + students[2].name);  
29    }  
30
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
```

```
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
```

```
Inside arrays ...
```

```
Mid-Term 1: 90
```

```
Mid-Term 2: 70
```

```
Final: 80
```

```
Project: 100
```

```
# exams: 4
```

```
Student 1: Student@681a9515
```

```
Student 2: Student@3af49f1c
```

```
Student 3: null
```

```
C:\javaindepth\src\com\semanticsquare\basics>
```