

3. Non-static variables:

1. Non-static variables can be accessed only after creating objects.
2. Non static variables are created outside all the methods but inside a class without static word
3. If you donot initialize non-static variables, then depending on the data type Auto initialization would take place

Example 1:

```
public class A {
```

```
    int i = 10;
```

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

```
        A a1 = new A();
```

```
        System.out.println(a1.i);
```

```
    }
```

```
}
```

Output:

10

Example 2:

```
public class A {
```

```
    int i;
```

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

```
        A a1 = new A();
```

```
        System.out.println(a1.i);
```

```
    }
```

```
}
```

Output:

0

Example 3:

How do i store mobile number in java which is 10 digit ?

//Memory size allocated for datatypes in java

```
public class A {
```

```
// Long Literal - long value
```

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

```
long i = 9632882052L;
```

```
System.out.println(i);
```

```
}
```

```
}
```

Example 4:

```
//Memory size allocated for datatypes in java
```

```
public class A {
```

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

```
float i = 10.3F;
```

```
System.out.println(i);
```

```
}
```

```
}
```

Output: 10.3

Example 5:

```
public class A {
```

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

```
        var i = 9632882052;
```

```
        System.out.println(i);
```

```
    }
```

```
}
```

Ouput: Error

Example 6:

```
public class A {
```

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

```
var i = 9632882052L;
```

```
System.out.println(i);
```

```
}
```

```
}
```

Output:

9632882052

Type Casting in Java:

Converting a particular datatype into required datatype is called as type casting

There are two types:

1. Auto upcasting:

Moving the data from smaller memory to bigger memory is called as auto upcasting

During upcasting data loss should not happen

Example 1:

```
public class A {
```

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

```
        byte i = 10;
```

```
        int j = i ;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```

Output:

10

Example 2:

```
public class A {
```

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

```
byte i = 10;
```

```
var j = i ;
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output:

10

Example 3:

```
public class A {
```

```
//Because long can store only integer values, 10.3 ----- 10
```

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

```
float i = 10.3f; //4
```

```
long j = i ;
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output: Error

Explicit Downcasting:

Moving the data from bigger memory to smaller memory is called as explicit downcasting

Example 1:

```
public class A {
```

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

```
        long i = 10;
```

```
        int j = i;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```


Output:

Error

Example 2:

```
public class A {
```

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

```
        long i = 10;
```

```
        int j = (int)i ;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```

Output:

10

Example 3:

```
public class A {
```

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

```
double i = 10.3;
```

```
float j = i ;
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output: Error

Example 4:

```
public class A {
```

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

```
double i = 10.3;
```

```
float j = (float)i ;
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output: 10.3

Example 5:

```
public class A {
```

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

```
        float i = 10.3F;
```

```
        long j = i ;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```

Output: Error

Example 6:

```
public class A {
```

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

```
        float i = 10.3F;
```

```
long j = (long)i ;
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output: 10 (value .3 the decimal part is lost)