



BITS Pilani
Pilani Campus

Object Oriented Programming CS F213

J. Jennifer Ranjani

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

Chamber: 6121 P, NAB

Consultation: Friday 4.00 p.m. – 5.00 p.m.



More on 'Static' Keyword

Static method



- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

Overloading Static Method

```
public class Main{  
    static int a ;  
    static float b;  
    static void assign(int A){  
        a = A;    }  
    static void assign(int A, float B){  
        a = A;  
        b = B;    }  
    public static void main(String []args){  
        Main.assign(10);  
        System.out.println("Values are: a = "+a+" b = "+b);  
        Main.assign(20,3.2f);  
        System.out.println("Values are: a = "+a+" b = "+b);}}
```

Static Block



- Used for initializing static variables.
- Static block will be executed when the class is loaded in the memory.
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

'final' Keyword



Final Variable	→	To create constant variables
Final Methods	→	Prevent Method Overriding
Final Classes	→	Prevent Inheritance



Mutable and Immutable Objects

Immutability and Instances



- Mutable Objects: Contents of an instance that can be modified.
- Eg: Immutable: `java.lang.String`
Mutable: Account
- When the contents of the String instance are modified, a new string object is created.

How to create an Immutable class?



- Class must be declared as final
 - So that child classes can't be created
- Data members in the class must be declared as final
 - So that we can't change the value of it after object creation
- A parameterized constructor
- Getter method for all the variables in it
- No setters
 - To not have option to change the value of the instance variable

Immutable Class - Example



```
final class Account{  
    final int acc;  
    final String name;  
    final float amount;  
  
    Account(int acc,String name,float amt){  
        this.acc = acc;  
        this.name = name;  
        this.amount = amt;  }  
  
    int getAcc(){  
        return acc;}  
  
    String getName() {  
        return name; }  
  
    float getAmount() {  
        return amount; }}
```

Immutable Class - Example



```
class TestAccount{  
    public static void main(String[] args) {  
  
        Account a= new Account(111,"Ankit",5000);  
  
        System.out.println("Acc: "+a.getAcc()+" Name: "+a.name);  
  
        a.amount = 1000;  
    }  
}
```

Output:

```
Exception in thread "main" java.lang.Error:  
Unresolved compilation problem:  
The final field Account.amount cannot be  
assigned
```



Arrays

BITS Pilani
Pilani Campus

Arrays



- Syntax to declare an array
 - `int[] arr;`
 - `int []arr;`
 - `int arr[];`
- Instantiation of an array
 - `arr = new int[size];`
- Arrays can be accessed using
 - Simple for loop
 - For each loop
 - Labelled for loop

For each loop

```
int arr[]={12,23,44,56,78};  
    //Printing array using for-each loop  
for(int i:arr){  
    System.out.println(i);  
}
```

Labelled For Loop



aa:

```
for(int i=1;i<=3;i++){  
    bb:  
    for(int j=1;j<=3;j++){  
        if(i==2&& j==2){  
            break bb;  
        }  
        System.out.println(i+" "+j);  
    }  
}
```

Copying a Java Array



public static void arraycopy(Object src, **int** srcPos, Object dest, **int** destPos, **int** length)

- arraycopy method of the System class is used to copy an array to another.

```
int a[] = {2,3,5};
```

```
int b[] = new int[a.length];
```

```
System.arraycopy(a, 1, b, 0, a.length-1);
```

```
for(int i=0;i<b.length;i++)
```

```
System.out.print(" "+b[i]);
```

Output:
3 5 0

Printing an Array using toString



```
import java.util.*;
public class Test{
    public static void main(String args[]){
        int arr[] = {1, 2, 3};
        System.out.println(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output:

```
[I@2a139a55
[1, 2, 3]
```

Array Class (import java.util.*)



static type	binarySearch(type[] a, type key) Searches the specified array of type for the specified value using the binary search algorithm.
static boolean	<u>equals</u>(type[] a, type[] a2) Returns true if the two specified arrays of type are equal to one another.
static void	<u>fill</u>(type[] a, type val) Assigns the specified type value to each element of the specified array of type.
static void	<u>fill</u>(type[] a, int fromIndex, int toIndex, type val) Assigns the specified type value to each element of the specified range of the specified array of types.
static void	<u>sort</u>(type[] a) Sorts the specified array of type into ascending numerical order.
static void	<u>sort</u>(type[] a, int fromIndex, int toIndex) Sorts the specified range of the specified array of type into ascending numerical order.
	type = byte, char, double, float, int, long, short, Object

Array Class - Example



```
int a[] = {2,3,5,1,4,7};
```

```
for(int i=0;i<a.length;i++)  
System.out.print(a[i]+" ");
```

```
System.out.println();  
Arrays.sort(a,0,4);  
System.out.println(Arrays.toString(a));
```

```
Arrays.sort(a);  
System.out.println(Arrays.toString(a));
```

```
System.out.println("Binary Search for 5 is "+Arrays.binarySearch(a, 5));
```

Output:

2 3 5 1 4 7

[1, 2, 3, 5, 4, 7]

[1, 2, 3, 4, 5, 7]

Binary Search for 5 is 4

Array Class - Example



```
int a[]= {2,3,5,1,4,7};
```

```
System.out.println(Arrays.toString(Arrays.copyOf(a, a.length)));
```

```
System.out.println(Arrays.toString(Arrays.copyOfRange(a, 1,4)));
```

```
Arrays.fill(a,4,a.length,1);
```

```
System.out.println(Arrays.toString(a));
```

```
Arrays.fill(a,1);
```

```
System.out.println(Arrays.toString(a));
```

Output:

```
[1, 2, 3, 4, 5, 7]
```

```
[2, 3, 4]
```

```
[1, 2, 3, 4, 1, 1]
```

```
[1, 1, 1, 1, 1, 1]
```

Predict the output

```
int arr1[] = {1, 2, 3};  
    int arr2[] = {1, 2, 3};  
    if (arr1 == arr2)  
        System.out.println("Same");  
    else  
        System.out.println("Not same");
```

Output: Not same

Problem: == compares the array references

Solution: Arrays.equals(arr1, arr2)