Making Calculator Program better

- We will try to make the calculator program better whatever we have studied so far.
- In case of division operation, there is a possibility that we input denominator as 0 which is an invalid scenario.
- As a better programmer, we should check such kind of scenario.
- We will see in the code how to do it.

Loops in Java

- Three basic ways to write loops in Java.
  - While loop
  - Do-While loop
  - For loop

While Loop

- Repeatedly executes a statements as long as the condition is true.
  - Here condition is checked at the start of loop
  - There is a possibility that statement may never execute at all if condition is not satisfied.
  - Syntax is

    while(some condition)

    Statement;
  - We will see a program to make it more clear.

Do-While loop

- Repeatedly execute a statements as long as the condition is true.
  - Condition is checked at the end of the loop
  - So even if the condition is not true in the starting itself, it will run the statement at least one time.
  - Syntax is

```
do
    statement;
    while(some condition);
```

  - We will see a program to make it more clear.

For Loop

- Repeatedly executes a statement as long as the condition is true.
  - Condition checked at the loop start
  - Provides simplified notation for loop control values
  - Syntax

    for(initialize, condition, update)

    Statements;
  - We will see a program to make it more clear. (Program to printing 1-10)

Array

- Array is a collection of similar type of elements that have contiguous memory location.
- **Java array** is an object that contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array..
  - Each element accessed via an index
  - Index range from 0 to number of elements minus 1
  - Number of elements can be found via array's length value

Array Contd..

- Advantage of Array in Java
  - Code Optimization: It makes the code optimized, we can retrieve or sort the data easily.
  - Random access: We can get any data located at any index position.
- Disadvantages of Array in Java
  - Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime.
  - To solve this problem, collection framework is used in java.

Array Contd..

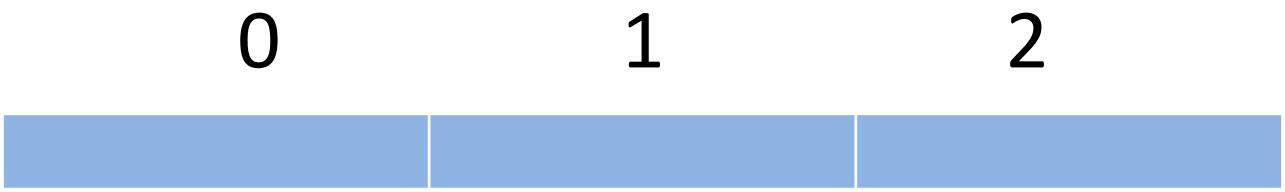
- Types of Array in Java
  - Single Dimension Array
  - Multi Dimension Array

Single Dimension Array

- Syntax to define Array

  float[] varName = new float[3];

- It is declared to store 3 variable of float data type.
- It will 1st allocate a memory space inside your computer



- We will see a program to make it more clear.

Multi Dimension Array

- In such case, data is stored in row and column based index (also known as matrix form).
- Syntax to define

```
float[][] varName = new float[3][1]; //one
                                    dimensional array only
float[][] varName = new float[3][3]; //3 rows 3
                                column, multi dimensional array
```

Multi Dimensional Array Contd..

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;
```

We will see one example to make it more clear.

Multi Dimensional Array Contd..

```
class Testarray3{
        public static void main(String args[]){
//declaring and initializing 2D array
                int arr[][]={{1,2,3},{4,5,6},{7,8,9}};
//printing 2D array
                for(int i=0;i<3;i++){
                        for(int j=0;j<3;j++){
                                System.out.print(arr[i][j]+" ");
                        }
                        System.out.println();
                }
        }
}
```

For each loop

- Executes a statements once for each member in an array
  - Handles getting collection length
  - Handles accessing each value
  - Syntax

  For(loop-varaible-declaration: array)

  statements;

Switch

- Transfer controls to a statements based on a value
  - Simplifies testing against multiple possible matches
  - Only primitive types supported as char and integers
  - A match can execute more than one statements
  - Use break to avoid “falling through”
  - Can optionally include default to handle any unmatched values

Switch contd..

- Syntax:

```
switch(test-value){
        case value-1:
        statements;
        case value-2:
        statements;
        .
        .
        .
        case value-n:
        statements;
        default:
        statements;
```

Summary

- Use the if-else statement to provide conditional logic
  - If-else statement can be chained together
- Block statements use brackets to group statements
  - Variables declared within a block are not visible outside the block
- Both while and do-while loops execute as long as condition is true.
  - The do-while loop body always execute at least once.
- The for loop provides simplified notation for loop initialization and control.
- For-each statement handles details of executing once for each array member.
- Switch statement simplifies notation of testing against multiple matches.

3. Agenda

- Classes
- Using Classes
- Classes as reference types
- Encapsulation and access modifier
- Basic Method
- Field accessors and mutators

Classes in Java

- Java is a n object-oriented language
- Objects encapsulate data, operations, and usage semantics
  - Allow storage and manipulation details to be hidden
  - Separates “WHAT” is to be done from “HOW” it is done
- Classes provide a structure for describing and creating objects

Classes in Java Contd..

- A class is a template for creating an object
  - Declared with the class keyword followed by class name
  - Java source file name normally has same name as the class
  - Body of the class is contained within brackets
  - A class is made up of both state and executable code
  - Fields
    - Store object state
  - Methods
    - Executable code that manipulates state and perform operations
  - Constructors
    - Executable code used during object creation to set initial state

Classes in Java Contd..

```
class flight{
        int passengers;
        int seats;
        Flight(){
                seats = 100;
                passengers = 0;
        }
        void add1Passenger(){
                if(passengers < seats)
                        passengers += 1;
        }
}
```

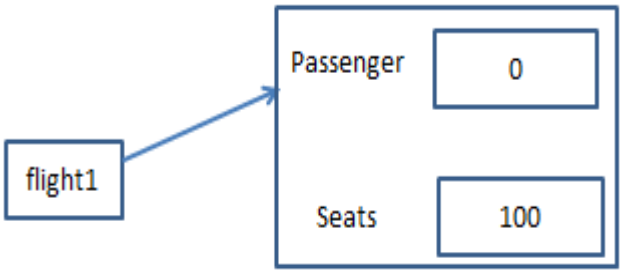
Using Classes

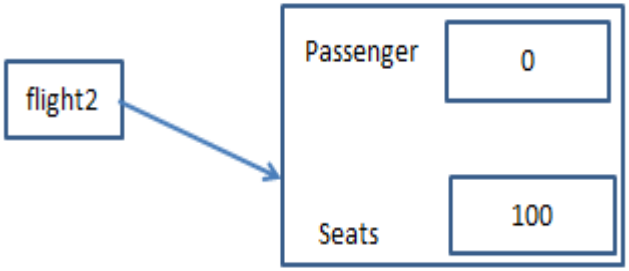
- Use the new keyword to create a class instance(a.k.a. an object)
  - Allocates the memory described by the class
  - Returns a reference to the allocated memory
- Syntax to declare

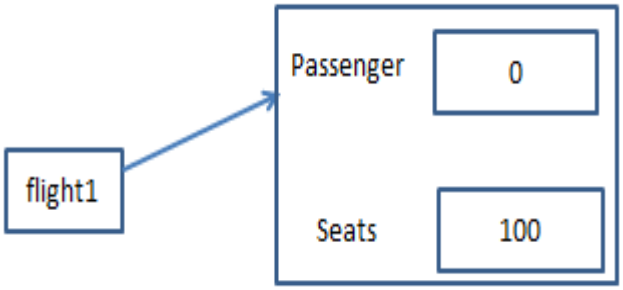
```
Flight nycToSf; //either we can declare in 2 line
nycToSf = new Flight(); //declare & create object
Flight slcToDallas = new Flight(); // or in 1 line
```

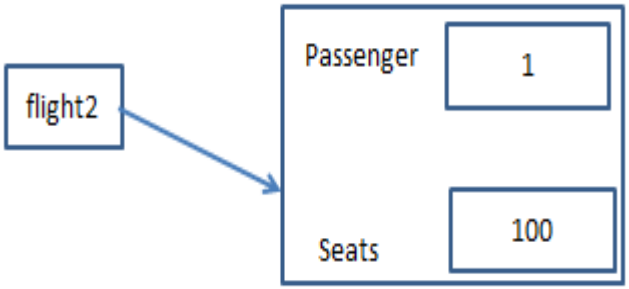
Classes are Reference Types

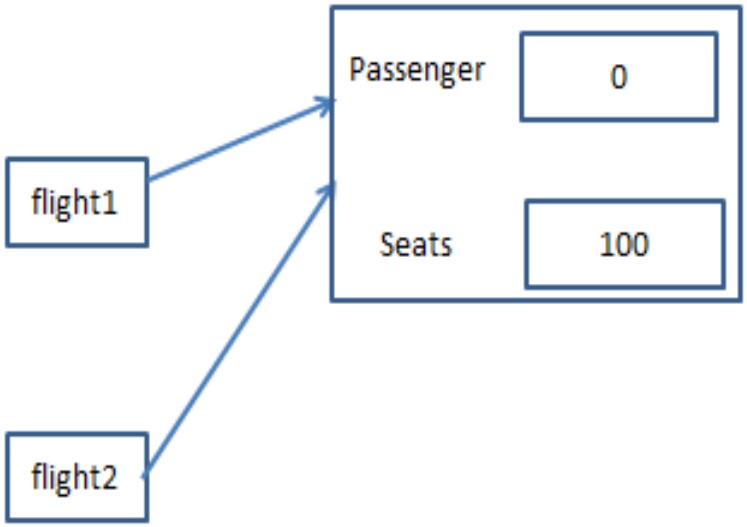
```
Flight flight1 = new Flight();
Flight flight2 = new Flight();

flight2.add1Passenger();
System.out.println(flight2.passengers);   // 1

flight2 = flight1;
System.out.println(flight2.passengers);   //0

flight1.add1Passenger();
flight1.add1Passenger();

System.out.println(flight2.passengers); //2
```

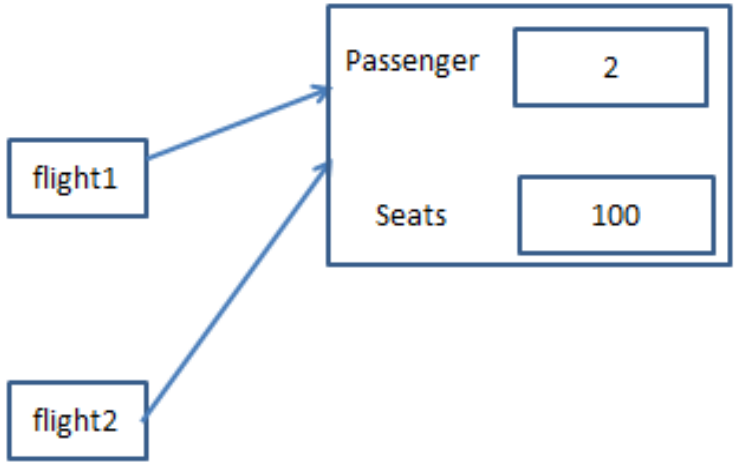

flight1
Passenger
0
Seats
100


flight2
Passenger
0
Seats
100


flight1
Passenger
0
Seats
100


flight2
Passenger
1
Seats
100


flight1
flight2
Passenger
0
Seats
100


flight1
flight2
Passenger
2
Seats
100

Points to be Noted

- When we assign one object to other, then we assign the references to the object and not the value.
- This is totally different with respect to the primitive data type where we copy the actual value to another, but here we just copy the reference of an object.

Encapsulation & Access Modifier

- Since Java is an Object Oriented language, we should use object oriented technique to built our program.
- The internal representation of an object is generally hidden. This means that external world(end-user) should be only familiar with what we are doing in the program not about how we are doing the program.
- This concept is called as encapsulation.
- Java uses access modifier to achieve encapsulation.

Basic Access Modifier

Modifier	Visibility	Usable on Classes	Usable on Members
No access modifier	Only within its own package(a.k.a. package private)	Y	Y
Public	Everywhere	Y	Y
Private	Only within its own class	N (as private applies to top-level classes, private is available to nested-classes)	Y

Applying Access Modifiers

```
public class flight{
    private int passengers;
    private int seats;
        public Flight(){
                        seats = 100;
                        passengers = 0;
            }
         public void add1Passenger(){
                        if(passengers < seats)
                                passengers += 1;
                        else
         private handleTooMany();
            }
            void handleTooMany(){
                    system.out.println(“Too Many”);
            }
}
```

```
Flight flight1= new Flight();
System.out.println(flight1.passengers);
/*not allowed as it will give compile time error as
access modifier used here is private */

flight1.add1Passenger();
/*No problem to call as it is public*/

flight1.handleTooMany();
/*not allowed as it will give compile time error as
access modifier used here is private */
```

Point to be Noted

- Normally class file has the same name as the name of the class.
- Once we mark the class as public, the source file name has to be the same name. This is mandatory, and so the previous class definition has to be in the file name called flight.java else it will throw an error.

Naming Classes

- Class name follow the same rules as variable name.
- Class name conventions are similar to variables with some differences.
  - Use only letters and numbers.
  - First character is always a letter
  - Follow the style often referred to as “Pascal Case”
    - Start of each word, including the first, is upper case
    - All other letters are lower case
  - Use simple, descriptive nouns
  - Avoid abbreviations unless abbreviation’s use is more common that full name.

Naming Classes

- Example:

  Class BankAccount{……}

  Class Person{……}

  Class TrainingVideo{……..}

  Class URL{…..}

Method Basics

- Executable code that manipulates state and perform operations
  - Name
    - Same rules and conventions as variables
    - Should be a verb or action
  - Return type
    - Mandatory to mention return type.
    - Use void when no value returned
  - Typed parameter list
    - Can be empty
  - Body contained with brackets

Method Basics

- Syntax

```
return-type name(typed-parameter-list){
        statements;
}

public class MyClass{
public void showSum(float x, float y, int count){
        float sum = x+y;
        for(int i =0; i <count; i++)
                system.out.println(sum);
        }
}

MyClass m = new MyClass();
m.showSum(7.5, 1.5, 2);
```

Exiting from a Method

- A method exits for one of three reasons
  - The end of the method is reached
  - A return statement is encountered
  - An error occurs
- Unless there's an error, control returns to the method caller.
- Example

Exiting from a method

```
public class MyClass{
    public void showSum(float x, float y, int count){
        if(count < 1)
                return;
        float sum = x+y;
        for(int i =0; i <count; i++)
                system.out.println(sum);
           return;
        }
}

    MyClass m = new MyClass();
    m.showSum(7.5, 1.5, 0);
    System.out.println("I am back");
```

Method Return Values

- A method returns a single values
  - A primitive value
  - A reference to an object
  - A reference to an array
    - Array are objects

Method Return Values

```
public class Flight{
        private int passenger;
        private int seats;
        //constructor and other method for more clarity
        public boolean hasRoom(Flight f2){
                int total = passenger + f2.passengers;
                if(total <= seats)
                        return true;
                else
                        return false;
        }
        public Flight createNewWithBoth(Flight F2){
                Flight newFlight = new Flight();
                newFlight.seats = seats;
                newFlight.passengers = passengers + f2.passengers;
                return newFlight;
        }
}
```