Introduction to Java:

- Java is a open source programming language created by Sun Microsystems overtaken by Oracle .
- **Java is a platform Independent**: – Platform independent means Java can run on any computer irrespective to the hardware and software dependency. Java does not depend on type of processor , RAM etc. Java will run on a machine which will satisfy its basic needs.
- Java is a secure Language
- Java is an Object Oriented Programming Language
- Collection of Open Source libraries:- Apache, Google, and other organization have contributed libraries, which makes Java development easy, faster and cost effective.

Features of Java:

Why Java?

•Mobile applications (specially Android apps)
•Desktop applications
•Web applications
•Web servers and application servers
•Games
•Database connection

Hello World Program :

For executing any Java program, the following software or application must be properly installed.

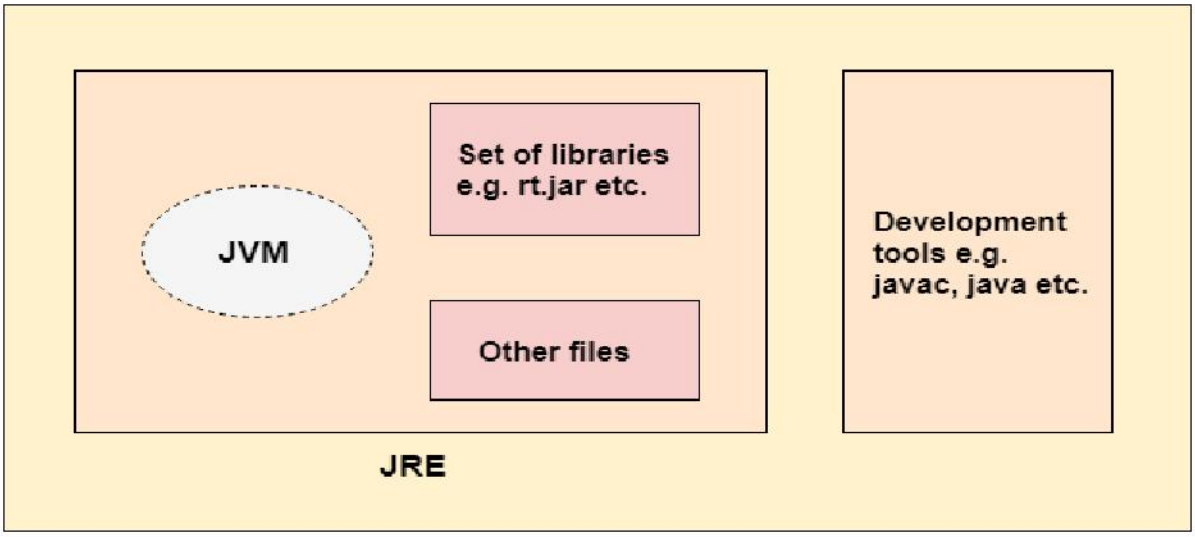
•Install the JDK if you don't have installed it.

•Set path of the jdk/bin directory

•Create the Java program

•Compile and run the Java program

JDK,JRE and JVM:



JDK : Java Development Kit

The Java Development Kit (JDK) is a software development environment which is used to develop Java applications .It physically exists. It contains JRE + development tools

JRE : Java Runtime Environment

The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

JVM : Java Virtual Machine

It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed

```
class Simple{

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

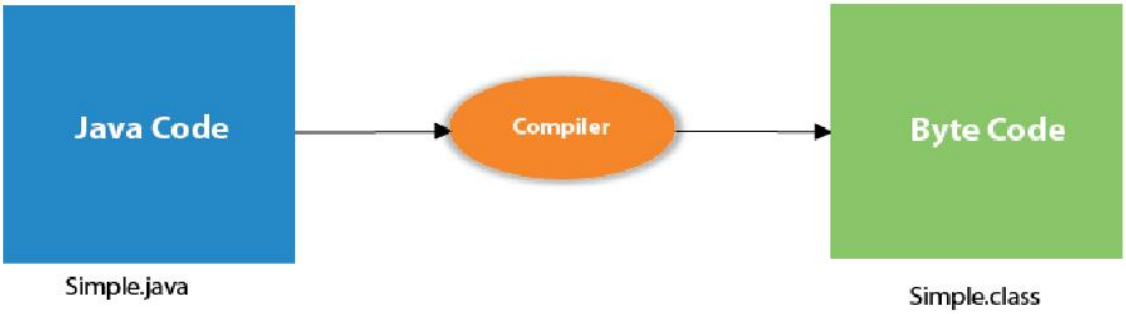
```
    System.out.println("Hello Java");
    }
}
```

To compile: javac Simple.java

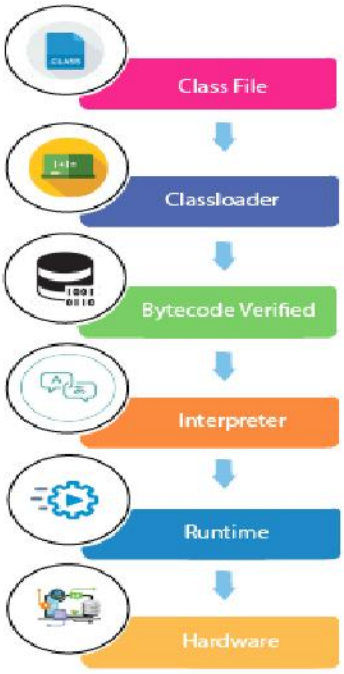
To execute: java Simple

Compilation Flow:

When we compile Java program using javac tool, the Java compiler converts the source code into byte code.



What happens in runtime?

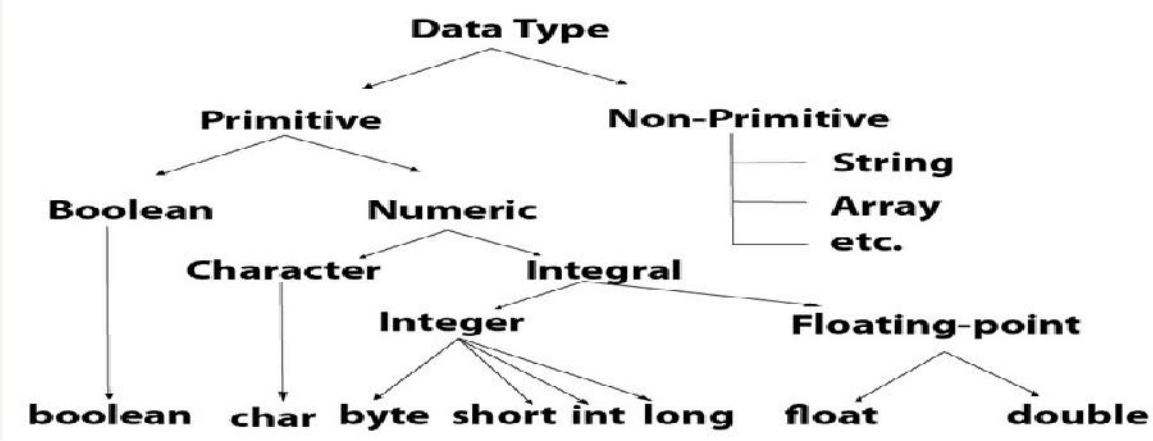


Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1.**Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.

2.**Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.



Java User Input :

Scanner :Scanner class is used to get user input, and it is found in the java.util package.
To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation

Note: If you enter wrong input (e.g. text in a numerical input), you will get an exception/error message (like "InputMismatchException").

Why does this issue occur?

This issue occurs because, when nextInt() method of Scanner class is used to read the age of the person, it returns the value 1 to the variable age, as expected. But the cursor, after reading 1, remains just after it.

```
abc
m
1_ // Cursor is here
xyz
pqr
```

Demo Code:

```
import java.util.Scanner;
import java.text.DecimalFormat;

public class Main
{
	public static void main(String[] args) {


		Scanner sc=new Scanner(System.in);

		int id=sc.nextInt();
		sc.nextLine();
		String name=sc.nextLine();
		char gender=sc.next().charAt(0);
		float marks1=sc.nextFloat();
		float marks2=sc.nextFloat();
		float marks3=sc.nextFloat();
		boolean active=sc.nextBoolean();
		double result=marks1+marks2+marks3;
		float result1=marks1+marks2+marks3;


		// Double will have more precision than float-More accurate
		System.out.println("Total result in double :"+result);

		//Float will have less precision value-Less accurate
		System.out.println("Total result in float :"+result1);


		//Display all the values in single Line with space and tab space using "\t"

		System.out.println(id+" "+name+" "+gender+" "+active+" "+result);
```

```
System.out.println(id+"\t"+name+"\t"+gender+"\t"+active+"\t"+result);


//Display all the values in different line using "\n"

System.out.println(id+"\n"+name+"\n"+gender+"\n"+active+"\n"+result);


//Display decimal values after specific digit


String s1=String.format("%.4f",result); //Display 4 decimal values

System.out.println("Display 4 decimal point:"+s1);

System.out.println("Display 2 Decimal point:"+String.format("%.2f",result));//Display
2 decimal values


//Round off


double x=89.976;

DecimalFormat dc=new DecimalFormat("#.#"); //90
System.out.println("Round off :"+dc.format(x));

DecimalFormat dc1=new DecimalFormat("#.##");//89.98
System.out.println("Round off:"+dc1.format(x));


//Type Conversion


//Automatic casting
int xy=10;

double y=xy;

System.out.println(xy);
System.out.println(y);

//Manual casting
double z=45.56;
int a=(int)z;

System.out.println(z);
System.out.println(a);
```

```
        double b=9/2;   //one of the operands would be float for integer divison
        System.out.println(b);

        double b1=9.0/2;
        System.out.println(b1);


        double b2=(double)9/2;
        System.out.println(b2);



            //Operator precedence

            double d1=5+3*3;//high priority for mutiplication

        System.out.println(d1);

        double d2=(5+3)*3; //high priority for parenthesis

        System.out.println(d2);

      //Mathematical Operations

        double avg=result/3;

        System.out.println("Average:"+avg);



    }
}
```

Sample Input:
67
Ashwa
Female
89.76
78.90
67.12
true

Sample Output:
Total result in double :235.77999877929688
Total result in float :235.78
67 Ashwa F true 235.77999877929688

67 Ashwa F true 235.77999877929688
67
Ashwa
F
true
235.77999877929688
Display 4 decimal point:235.7800
Display 2 Decimal point:235.78
Round off :90
Round off:89.98
10
10.0
45.56
45
4.0
4.5
4.5
14.0
24.0
Average:78.5933329264323

```
        }
    }
```

Conditional Operations

If **I reach home on time** AND **internet is working,** I will watch the Cricket match.

condition 1 = reaching home on time (True)

condition 2 = internet working (True)

result = watch cricket match (True)

True AND(&&) True = True

True && False = False

False && False = False

OR

If **I am offered a Java based project** OR I am **offered a Python based project,** I will accept it.

condition 1 = offered a Java based project (True)

condition 2 = offered a Python based project (True)

result = accept the project (True)

True OR(||) True = True

True || False = True

False || False = False

```
public class MyClass {
   public static void main(String args[]) {
     boolean x=true;
     boolean y=true;
     boolean z = false;

     System.out.println(x&&y);   //true AND true
     System.out.println(x&&z);   //true AND false

     System.out.println(x||y);   //true OR true
     System.out.println(x||z);   // true OR false

     System.out.println(x&&y?"TRUE":"FALSE");       //ternary operator
   }
}
```

```
public class MyClass {
   public static void main(String args[]) {
     int x= 10;
     int y = 15;
     int z = 10;

     System.out.println(x==z && y==15);   //true AND true
     System.out.println(x==z && x==y);   //true AND false

     System.out.println(x==z||y==15);   //true OR true
     System.out.println(x==z||x==y);   // true OR false

     System.out.println(x==10?"Yes":"No");       //ternary operator
   }
}
```

```
public class MyClass {
   public static void main(String args[]) {
     int x= 10;
     int y = 15;
     int z = 10;

     if(x==z && y==15)
```

```
    System.out.println("returns True");   //true AND true
    else
    {
       System.out.println("returns False");
    }
   }
}
```

```
public class MyClass {
   public static void main(String args[]) {
    int x= 10;
    int y = 15;
    int z = 10;

    if(x==z && y!=15)
    System.out.println("returns True");   //true AND true
    else
    {
       System.out.println("returns False");
    }
   }
}
```

```
public class MyClass {
   public static void main(String args[]) {
    String x = "Hello";   //string literal
    String y = "hello";   //string literal
    String z = new String("hello");    //string object

    System.out.println("Value of z is : "+z);

    if(x=="Hello")
      System.out.println("x is equal to Hello");
    else
      System.out.println("x is NOT equal to Hello");
   //-----------------------------------------------
    if(x==y)
      System.out.println("x is equal to y");
    else
      System.out.println("x is NOT equal to y");
   //-----------------------------------------------
    if(y==z)
      System.out.println("y is equal to z");
    else
      System.out.println("y is NOT equal to z");
   //-----------------------------------------------
   if(y.equals(z))
      System.out.println("y is equal to z");
```

```
    else
      System.out.println("y is NOT equal to z");

  }
}
```

Practice:

Write a Java code to declare 2 integers and assign some value to them. Check whether the two integers are equal.

1) Using If Else
2) Using ternary operator

Write a Java code to declare two strings objects and assign some value to them. Check whether both are equal.

Write a Java code to declare 2 boolean variables b1 as true and b2 as false. Print result using && and || between b1 and b2.

Java Basics

Basic Logical Operations

4th Feb 2022

Topics

- Introduction
- Relational Operators(Recap)
- Logical Operators
- Examples
- Test your Knowledge
- Summary
- Q/A

Introduction

Consider the below statements

1. Age greater than 18?
2. You can apply for scholarship if you have scored minimum 80% for both English **and** Physics **and** scored minimum 90% in Mathematics.
3. You can apply for programmer job if you have proficiency level above 3 in Java **or** Python **or** C. (proficient in any of the language)

- The first statement is a relational expression, it can be denoted as below

  IF age > 18
- Second and third statements contain multiple conditions/expressions connected using AND or OR construct.
  - The second statement can be represented as below

  IF(eng_score >= 80 AND phy_score >= 80 AND math_score >= 90)

  THEN apply for scholarship.
  - The third statement can be represented as below

  IF(python_level > 3 OR java_level > 3 OR c_level >3)

  THEN apply for programmer job.

Relational Operators

Relational operators are used to find the relation between two variables.

Operator	Description	Example
== (equal to)	Returns true if the value of both operands are equal else return false.	x == y
!= (not equal to)	Returns true if the value of both operands are not equal else return false.	x != y
> (greater than)	Returns true if the value of left operand is greater than the value of right operand else return false.	x > y
< (less than)	Returns true if the value of left operand is less than the value of right operand else return false.	x < y
>= (greater than or equal to)	Returns true value of left operand is greater than or equal to the value of right operand else return false.	x >= y
<= (less than or equal to)	Returns true if the value of left operand is less than or equal to the value of right operand else return false.	x <= y

Relational Operator :Example1

- Read marks of 3 subjects(out of 100) of a student and check if average marks is greater than or equal to 80%.

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    // read 3 scores
    int score1 = s.nextInt();
    int score2 = s.nextInt();
    int score3 = s.nextInt();

    int avgScore = (score1 + score2 + score3) / 3;

    if(avgScore >= 80){
        System.out.println("avg score is greater that or equal to 80");
    }
    else{
         System.out.println("avg score is not greater that or equal to 80");
    }
}
```

Relational Operators :Test Your Knowledge Q1

- What is the output of the below program?

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

    int x = 10;
    int y = 20;
    int z = 20;

    System.out.println(x == y);
    System.out.println(y != z);
    System.out.println( x < y);
    System.out.println( y < z );
    System.out.println( y >= z);
    System.out.println( x <= y);

}
```

Logical Operators

- Logical operators AND(&&) and OR(||) checks the value of the operands(conditions/expressions) in the left and right side of the operator and return either true or false depends on the value of the operands.
- The conditions or expressions are mainly any relational expression in Java.

Syntax : (condition1) operator(condition2) *[operator(condition3).....]*

Operator	Name	Description	Example
&&	Logical AND	Returns **true** if both conditions are true else return false.	(age <= 30) **&&** (work_experience >= 4)
||	Logical OR	Returns **true** if any of the condition is true else return false.	(age < 5) || (age >60)
!	Logical NOT	Returns **true** if the condition is false and returns **false** if the condition is true.	!(condition) !(speed < 100)

Logical Operators(cont.)

Value of Left Side Condition/Expression	Operator	Value of Right Side Condition/Expression	Result
TRUE	**&&**	TRUE	**TRUE**
TRUE	**&&**	FALSE	**FALSE**
FALSE	**&&**	TRUE	**FALSE**
FALSE	**&&**	FALSE	**FALSE**
TRUE	**||**	TRUE	**TRUE**
TRUE	**||**	FALSE	**TRUE**
FALSE	**||**	TRUE	**TRUE**
FALSE	**||**	FALSE	**FALSE**

Table that demonstrates the result of logical operation based on different conditions.

Example Program 1

Read the body temperature of a person for last 3 days and check if the temperature is above 37 degrees Celsius for last 3 days.

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    // read temperature
    double temp1 = s.nextDouble();
    double temp2 = s.nextDouble();
    double temp3 = s.nextDouble();

    if(temp1 >37 && temp2 >37 && temp3 >37 ){
        System.out.println("temp is above 37 in all three days");
    }
    else{
         System.out.println("temp is not above 37 in all three days");
    }
}
```

Example Program2

- Read the body temperature of a person for last 3 days and check if the temperature is above 37 degrees Celsius in any of the three days.

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    // read temperature
    double temp1 = s.nextDouble();
    double temp2 = s.nextDouble();
    double temp3 = s.nextDouble();

    if(temp1 >37 || temp2 >37 || temp3 >37 ){
        System.out.println("temp is above 37 degrees in one or more day");
    }
    else{
         System.out.println("temp is not above 37");
    }
}
```

Example Program3

- A customer wants to buy a new tablet PC online based on the below criteria.
  - Screen size should be 10 inch and above
  - Memory(RAM) capacity should be 4GB and above
  - Price should be less than 40000 INR
  - Maker should be any of the below
    - Samsung
    - Apple
    - Nokia

- Write a program to read the size, memory, price and maker data and check if it meets the criteria listed above.

Example Program3 - Solution

```
public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  double screenSize = scanner.nextDouble();//read screen size
  int ramCapacity = scanner.nextInt(); //read ram capacity
  double price = scanner.nextDouble(); // read price
   scanner.nextLine();
  String maker = scanner.nextLine(); // read maker

  System.out.println(screenSize+" "+ramCapacity+" "+price+" "+maker);

  if( (screenSize >= 10)&&(ramCapacity>=4)&&(price <40000) &&
          ( maker.equals("Samsung")|| maker.equals("Apple")||maker.equals("Nokia")) ){
      System.out.println("Select Product");
  }
  else{
      System.out.println("Reject Product");
  }
}
```

Logical Operators : Test Your Knowledge- Q1

What will be the output of the below program?

```
public static void main(String[] args) {
    int counter = 0;

    if (counter > 0 && ++counter > 0) {
        System.out.println("True");
    }

    System.out.println("Counter = " + counter);
}
```

Q1- Answer & Explanation

- The logical AND(&&) and OR(||) operators exhibit short-circuit behavior.
  - The second operand(condition) is evaluated only when the result of the conditional operator cannot be deduced solely by evaluating the first operand.

- While using the &&(AND) operator, if the first operand(expression/condition) is false then it will not evaluate the second operand(expression/condition).
  - (Condition1 is **false**) && (Condition2)

- While using the ||(OR) operator, if the first operand(expression/condition) is true then it will not evaluate the second operand(expression/condition).
  - (Condition1 is **true**) || (Condition2)

Q1- Answer & Explanation

What will be the output of the below program?

```
public static void main(String[] args) {
    int counter = 0;
          Condition1          Condition2
    if (counter > 0 && ++counter > 0) {
        System.out.println("True");
    }

    System.out.println("Counter = " + counter);
}
```

Condition 1 is false so condition 2 will not be evaluated (executed) so the value of counter will remain same.

Result:
Counter = 0

Test Your Knowledge- Q2

What will be the output of the below program?

```
public static void main(String[] args) {
    int counter = 0;

    if (counter >= 0 && ++counter >= 0) {
        System.out.println("True");
    }

    System.out.println("Counter = " + counter);
}
```

Test your Knowledge – Q3

- What will be the output of the below program?

```
public static void main(String[] args) {
    int counter = 0;

    if (counter >= 0 || ++counter >= 0) {
        System.out.println("True");
    }

    System.out.println("Counter = " + counter);
}
```

Summary

- Logical operators AND(&&) and OR(||) are used to deduce a Boolean result(true or false) after checking the result of two conditions/expression.
- Logical AND(&&) operator returns true if both conditions are true.
- Logical OR(||) operator returns true if any of the condition is true.
- Both AND(&&) and OR(||) are short circuiting operators, both operators won't evaluate the second condition/expression if the final result can be deduced by evaluating the first condition/expression.

Java Basics

Numeric Logic Building

11 Feb 2022

Topics

- Arithmetic and Assignment Operators
- Looping / Iteration
- Problems and Solutions
- Test Your Knowledge

Arithmetic and Assignment Operators

Operators	Description	Example (let x =10 and y =3)
+ and -	Addition and subtraction	x + y => 13 x – y => 7
* and /	Multiplication and Division	x * y => 30 x / y => 3
%	remainder after division (modulo division)	x % y => 1
+= and -=	Addition assignment and Subtraction assignment	x += y => x = x + y => 13 x -= y => x = x - y => 7
*= and /=	Multiplication assignment and Division assignment	x *= y => x = x * y => 30 x /= y => x = x / y => 3
%=	Modulo assignment	x %= y => x = x % y => 1

Looping / Iteration

- Repetition or looping means executing a single or group (block) of code more than once if the given condition is true.

```
for( initial value ;  condition ;  increment){
// code
}
```

```
while(condition){
// code
}
```

```
do{
 //code
while(condition);
```

Looping / Iteration

```
for( initial value ;  condition ;  increment){
// code
}
```

```
int sum = 0;
for( int i=1;   i <10 ;   i+=2){
    sum += i;
}
```

```
while(condition){
// code
}
```

No Output?

```
int x = 10;
while(x < 10){
    System.out.printf("x is %d",++x);
}
```

```
do{
 //code
while(condition);
```

Prints y is 11

```
int y = 10;
do{
    System.out.printf("y is %d",++y);
}while(y < 10);
```

Problem1

- Print all odd numbers in a range (read the starting number and ending number)

- Example
  - Input : 1 and 10
  - Output : 1 3 5 7 9

Problem1-Solution

- Print all odd numbers in a range (read the starting number and ending number)

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    // read the range
    int start = s.nextInt();
    int end = s.nextInt();
    System.out.println("Printing odd numbers from "+start+ " to "+end);
    //use a loop to print odd numbers
    // n%2== 0 means n is even
    for(int i=start; i<=end ;i++){
            if(i%2 != 0)
            System.out.println(i);
    }

}
```

Problem 2

- Read two number and find the value of one number raised to the power of another.

- Example
  - Input : 2 and 5
  - Output : 32 => (2^5 => 2*2*2*2*2)

Problem 2 -Solution

- Read two number and find the value of one number raised to the power of another.

```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    //read x and y
    int x = scan.nextInt();
    int y = scan.nextInt();
    int result=1;
    // x^y
    for(int i=0;i<y;i++){
     result = result*x;
    }

    System.out.println(x+"^"+ y+" = "+result);
}
```

Problem3

- Read a number and print all digits in that number.
  - Input : 12345
  - Output : 5 4 3 2 1

Problem3 - Solution

- Read a number and print all digits in that number.

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

    int n = 1234567;

    /*
    Extract the last digit of the number n by n%10
    Update the value of n by n/10 and repeat the above step
       till n is not equals to 0.
    */

    while(n > 0){
      int digit = n%10;
      System.out.println(digit);
      n /= 10;
    }

}
```

Problem 4

- Read n number using for loop and find the biggest number, smallest number and average of all numbers.

- Example - 5 numbers
  - Input : 3 5 7 2 1
  - Output : Biggest Number = 7

    Smallest Number = 1

    Average = 3

Problem 4 - Solution

- Read n number using for loop and find the biggest number, smallest number and average of all numbers.

```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    //how many numbers
    int count = scan.nextInt();
    int big=0, small=0, sum =0, avg =0;

    for(int i=0 ; i<count;i++){
      int n = scan.nextInt(); // read each number
        if(i==0)small=n; // initially set the first number as smallest

        if(n > big)big =n;
        if(n<small)small=n;
        sum +=n; // sum = sum + n
    }

     System.out.println("biggest number :"+big+" smallest number: "+small+" average: "+sum/count);
}
```

Problem 5

- Read an amount and split that into 1 rupee , 10-rupee, 100-rupee , 500-rupee and 2000-rupee denominations.

- Example
  - Input 12345
  - Output : 6-2000 0-500 3-100 4-10 5-1

Problem 5 - Solution

```
public static void main(String[] args) {
    int amount = 25561;
    int denomination_count =5; // 1, 10 , 100 , 500 , 2000 -- can add more
    //declare variables to store denomination count
    int _2000Count=0, _500Count=0 ,_100Count=0,_10Count =0,_1Count =0;

  //divide amount by each denomination and reduce count*denomination from amount
  for (int i = 0; i < denomination_count; i++) {
      if (amount >= 2000) {
          _2000Count = amount / 2000;
          amount = amount - _2000Count * 2000; }
      else if (amount >= 500) {
          _500Count = amount / 500;
          amount = amount - _500Count * 500;}
      else if (amount >= 100) {
          _100Count = amount / 100;
          amount = amount - _100Count * 100;}
      else if (amount >= 10) {
          _10Count = amount / 10;
          amount = amount - _10Count * 10;}
      else
          _1Count = amount ;
  }//end for

      //print result
      System.out.println(_2000Count+ "-2000 ");
      System.out.println(_500Count+"-500 ");
      System.out.println(_100Count+"-100 ");
      System.out.println(_10Count+"-10 ");
      System.out.println(_1Count+"-1");
}
```

Test Your Knowledge

What will be output of the below code snippet?

```
int x = 1234567;
int c = 0;
while(x > 0){
    x /=10;
    c++;
}
System.out.println(c);
```

Prints the number of digits in the given number.
Output : 7

Test Your Knowledge

- What will be the output of below program?

```
int number = 12345;
int result = 0;
do {
    result *= 10;
    result += number % 10;
    number /= 10;
} while (number > 0);

System.out.println("Result "+result);
```

Reverse the given number.

Output : Result 54321

Test Your Knowledge

- What will be the output of the below program?

```
for(int i=1;i<=10;i++){
    for(int j=1;j<=10;j++){
        System.out.printf("%2d ",i*j);
    }
    System.out.println("");
}
```

Prints the multiplication table.

Output:

```
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
   -----
```

Thank You!

Try It Yourself -1

- Read a number and print sum of all digits in that number.

- Example
  - Input : 2030
  - Output : 5

Try It Yourself -2

- **Calculate Average score of all students in each batch**

  There are N batches of students in a course and there are M students in each batch.

Read N and M and then read the total score of each student(out of thousand) in each batch.

Calculate the average score of each batch and print it.

(use nested for loop)

- Example
  - Input : N -> 2 , M–> 5

    700 800 600 500 900

    900 700 600 950 860

    Output : batch1 average -> 700 batch2 average -> 802

Logic Building with String manipulation – Java 09Feb 2-3PM

1. **String concatenation**

```
public class Main{
   public static void main(String args[])
   {
String str1= "Welcome";
String str2 = "Java session";
String f = str1+" to "+str2;
System.out.println(f);
} }
```

2. **toLowerCase() and toUpperCase()**

```
String str1= "Welcome";
System.out.println(str1.toLowerCase());
System.out.println(str1.toUpperCase());
```

3. **substring()**

```
String str2 = "Java session";
String newstr = str2.substring(5,12);
System.out.println(newstr);
```

4. **trim()**

```
String str2 = "Java session  ";
System.out.println(str2);
System.out.println(str2.trim());
```

5. **Reverse string using charAt()**

```
String str2 = "Welcome";
String rev = "" ;
for(int i = str2.length()-1; i>=0; i--)
{
  rev = rev + str2.charAt(i);
 }
System.out.println(rev);
```

6. **indexOf()**

```
String str2 = "Welcome";
System.out.println(str2.indexOf("e",2));
```

Find count of vowels in a String

```
String str = "Good Afternoon";
int c=0;
 for (int i = 0; i < str.length(); i++)
{
        if(str.toLowerCase().charAt(i)=='a' || str.toLowerCase().charAt(i)=='e' ||
str.toLowerCase().charAt(i)=='i' || str.toLowerCase().charAt(i)=='o' ||
str.toLowerCase().charAt(i)=='u')
{
c=c+1;
}}
System.out.println(c);
```

Count of words in a sentence(String)

```
String str = "This is a Java code compiler ";
int c=1;
String news = str.trim();
 for (int i = 0; i < news.length(); i++)
{
        if(news.charAt(i)==' ')
{
c=c+1;
}}
System.out.println(c);
```

Odd/Even positioned substring

```
String str2 = "Digital";

    for(int i=0;i<str2.length();i=i+2)
    {
            System.out.println(str2.substring(i,i+1));
    }
```

Reading date as ddmmyyyy and displaying day, month and year

```
    String str = "12122013";

    String day = str.substring(0,2);
    String month = str.substring(2,4);
```

```
    String year = str.substring(4,8);

     System.out.println(day);
     System.out.println(month);
     System.out.println(year);
```

```
String str = "Happy New Year";

     // Creating array of string length
     char[] ch = new char[str.length()];

     for (int i = 0; i < str.length(); i++) {
        ch[i] = str.charAt(i);

           System.out.println(ch[i]);
        }

  }}
```

Reverse a string word by word

```
String str = "Happy New Year";

     String[] ns = str.split(" ");
     String nstr="";

     for(int j=0;j<ns.length;j++){

        for (int i = ns[j].length()-1; i >=0; i--)
        {

           nstr = nstr + ns[j].charAt(i);

        }
        nstr=nstr+" ";

     }
     System.out.println(nstr);
```

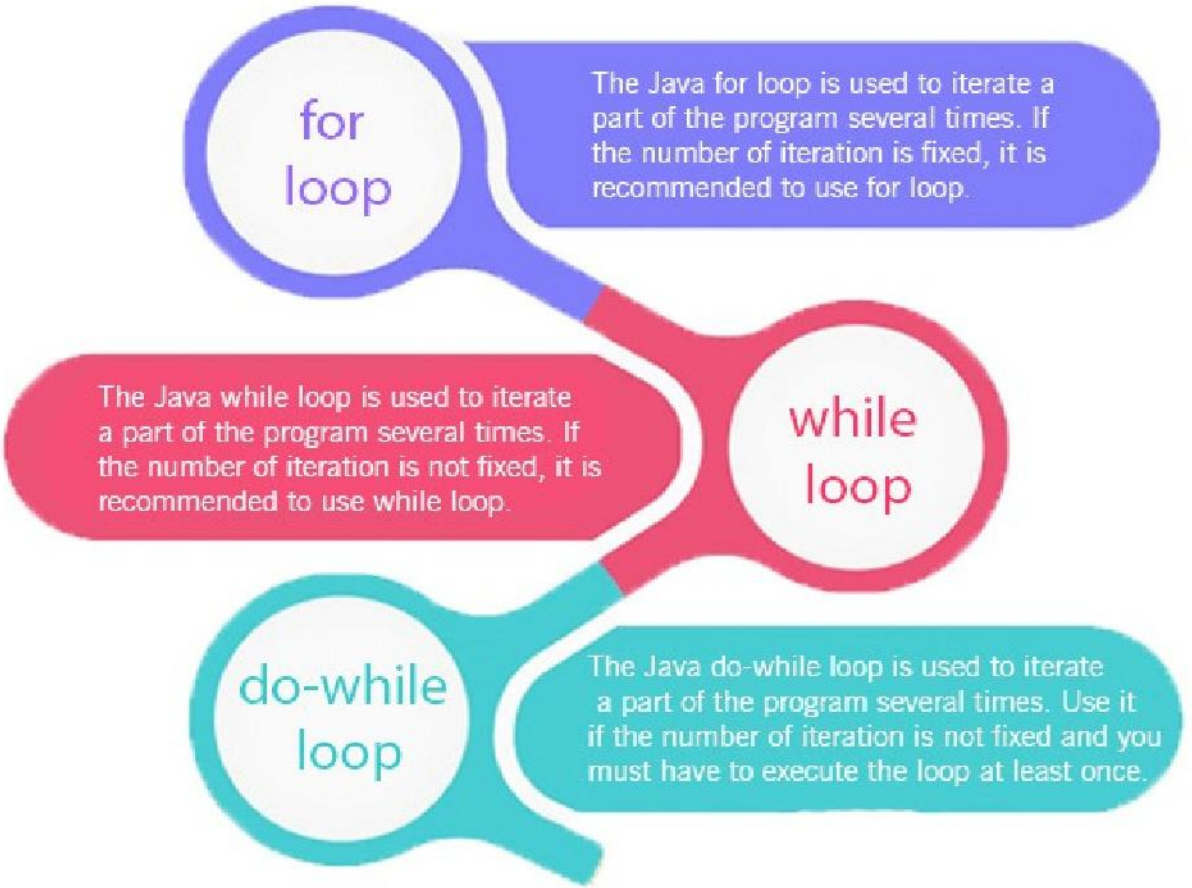
TATA CONSULTANCY SERVICES
Experience certainty.

Topic:Java Iterations

LOOPS

- Loops are used for automatically repeating similar task many number of times.
- Loops in java:
    - While
    - Do-While
    - For
    - Enhanced for

LOOPS


for loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.
while loop
The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
do-while loop
The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.

FOR LOOP

For loops execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
The syntax of a for loop is:

```
public class Loops {

   public static void main(String[] args) {
      //for(Initialization;Condition;Increment/Decrement)

       for(int i=1;i<=5;i++){
           System.out.print(i);
       }
   }
   }
```

OUTPUT:
12345

WHILE LOOP

The while statement continually executes a block of statements while a particular condition is true.

The syntax can be expressed as:

```
public class Loops {
public static void main(String[] args) {
        int i=1;
        while(i<=5){
            System.out.print(i);
          i++;
        }
} }
```

OUTPUT:

12345

DO WHILE LOOP

The difference between do-while and while is that do-while evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once. The syntax can be expressed as:

```
public class Loops {
   public static void main(String[] args) {
       int i=5;
          do{
              System.out.print(i);
          i++;
          }while(i<5);
      }  }
OUTPUT:
5
```

FOR EACH LOOP

Its simple structure allows one to simplify code by presenting for-loops that visit each element of an array/collection without explicitly expressing how one goes from element to element.

The syntax can be expressed as:

```
public class Loops {
   public static void main(String[] args) {
       int arr[]={2,9,4,5,7};
       for(int i:arr){



           System.out.print(i);
       }
   } }
```

Break

Break statement can also be used to jump out of a loop.

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

        for (int i = 0; i < 10; i++) {
        if (i == 4) {
        break;
        }
  System.out.println(i);
}

}
```

Continue

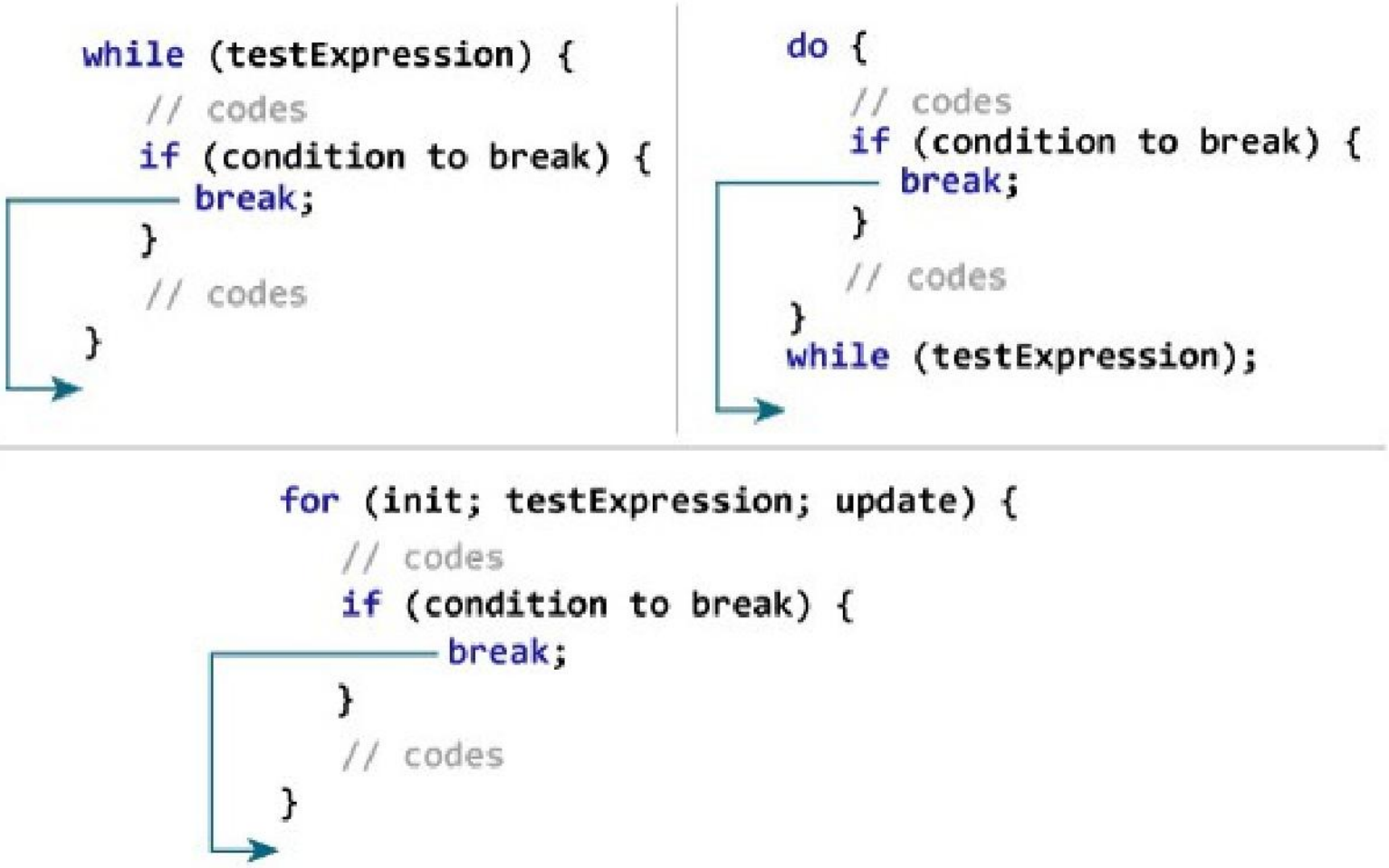
The Java continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

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

        for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);

}
```

Looping Statements



Assignments-Iterations

1. Write a Java Program to print Even Numbers from 0 to 20 using For loop
2. Write a Java Program to print the Numbers from 20 to 1 using While Loop
3.Write a Java Program to print the Odd Numbers from 0 to 50

Assignments-Basic Data Types

1.Write a Java program to read a string from the console and print each character of the string on a separate line.
2.Write a Java program to read a float value as an input and round it to 2 decimal digits.
3.Write a Java program to read a double value from console and print a number with 4 digits after the decimal point

Demo Code

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


	 //For Loop
	    for(int i=0;i<5;i++)
	        {
	            System.out.println("For Loop:"+i);
	        }

	     for(double j=0;j<5;j++)
	        {
	            System.out.println(j);
	        }

	    for(int k=1;k<=10;k++)
	        {
	            System.out.println(k);
	        }
```

```
//Reverse order from 5 ..1
        for(int i=5 ;i>0;i--)
            {
                System.out.println("Reverse Order :"+i);
            }

         //incrementing i by 2
        for(int i=0;i<=10;i=i+2)
            {
                System.out.println(i);
            }


        //Even numbers from 10 to 20

            for(int i=10;i<=20;i++)
            {
                if(i%2==0)
                {
                    System.out.println("Even Numbers:"+i);
                }
            }
```

```
//odd numbers from 10 to 20

        for(int i=10;i<=20;i++)
        {
            if(i%2!=0)
            {
                System.out.println("Odd Numbers:"+i);
            }
        }


       //While loop
      int i=1;
        while(i<5)
        {
            System.out.println("While loop :"+i);
            i++;
        }


        //Infinite while loop

        /*
        while(true)
        {
            System.out.println("Infinite loop");
        }*/
```

```
//Do while loop
    int b=0;

    do{
        System.out.println("Do while loop:"+b);
        b++;

    }while(b<5);


    //Do while loop
    int c=5;

    do{
        System.out.println("Do while loop:"+c);
        c++;

    }while(c<5);
```

```
//Break
    for(int l=0;l<5;l++)
   {
       if(l==4)
       {
           break;
       }

       System.out.println("Break:"+l);
   }


   for(int m=100;m>=10;m--)
   {

       System.out.println(m);
       if(m==99)
       {
           break;
       }


   }

       System.out.println("Out of loop");
```

```
//Continue
 for(int k=1;k<10;k++)
   {

      if(k==5)
      {
         continue;
      }
     System.out.println("Continue:"+k);


   }

     //For each loop
          int x=10;
          int y=90;
          int z=100;

          int arr[]=new int[5];
          arr[0]=10;
          arr[1]=30;
          arr[2]=90;
          arr[3]=89;
          arr[4]=67;

       System.out.println("Length of an array:"+arr.length);
       for(int a=0;a<arr.length;a++)
       {
          System.out.println(arr[a]);
```

```
//for each
        //for(type var:array)

        for(int xy :arr )
        {

            System.out.println("For Each loop:"+xy);
        }

    }
}
```

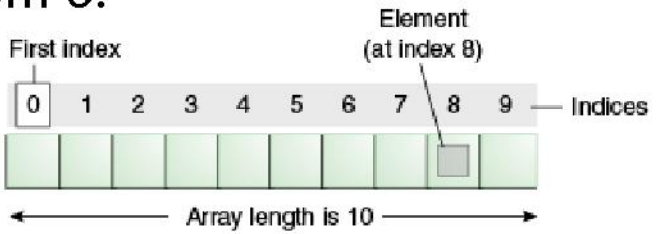
TATA CONSULTANCY SERVICES
Experience certainty.

Topic:Basic collections - Array & List basics

Array

Java array is an object containing fixed set elements of similar data type.

Array index starts from 0.



Array can be declared in the following ways:

```
int a[]={33,3,4,5};//declaration, instantiation and initialization

int a[]=new int[5];//declaration and instantiation

        a[0]=10;//initialization

        a[1]=20;

        a[2]=70;

        a[3]=40;

        a[4]=50;
```

Array Demo

```
class Testarray{
    public static void main(String args[]){
        int a[]=new int[5];//declaration and
                    instantiation
        a[0]=10;//initialization
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;
    //printing array
    for(int i=0;i<a.length;i++)//length is the
                        property of array
        System.out.print(a[i]);
        }    }
```

OUTPUT:

10 20 70 40 50

Advantages of Array

Advantages of Array:

- It is capable of storing many elements at a time.
- It allows random accessing of elements i.e. any element of the array can be randomly accessed using indexes.

Disadvantages of Array

Disadvantages of Arrays:

- Predetermining the size of the array is a must.
- There is a chance of memory wastage or shortage.
- To delete one element in the array, we need to traverse throughout the array.

Collections

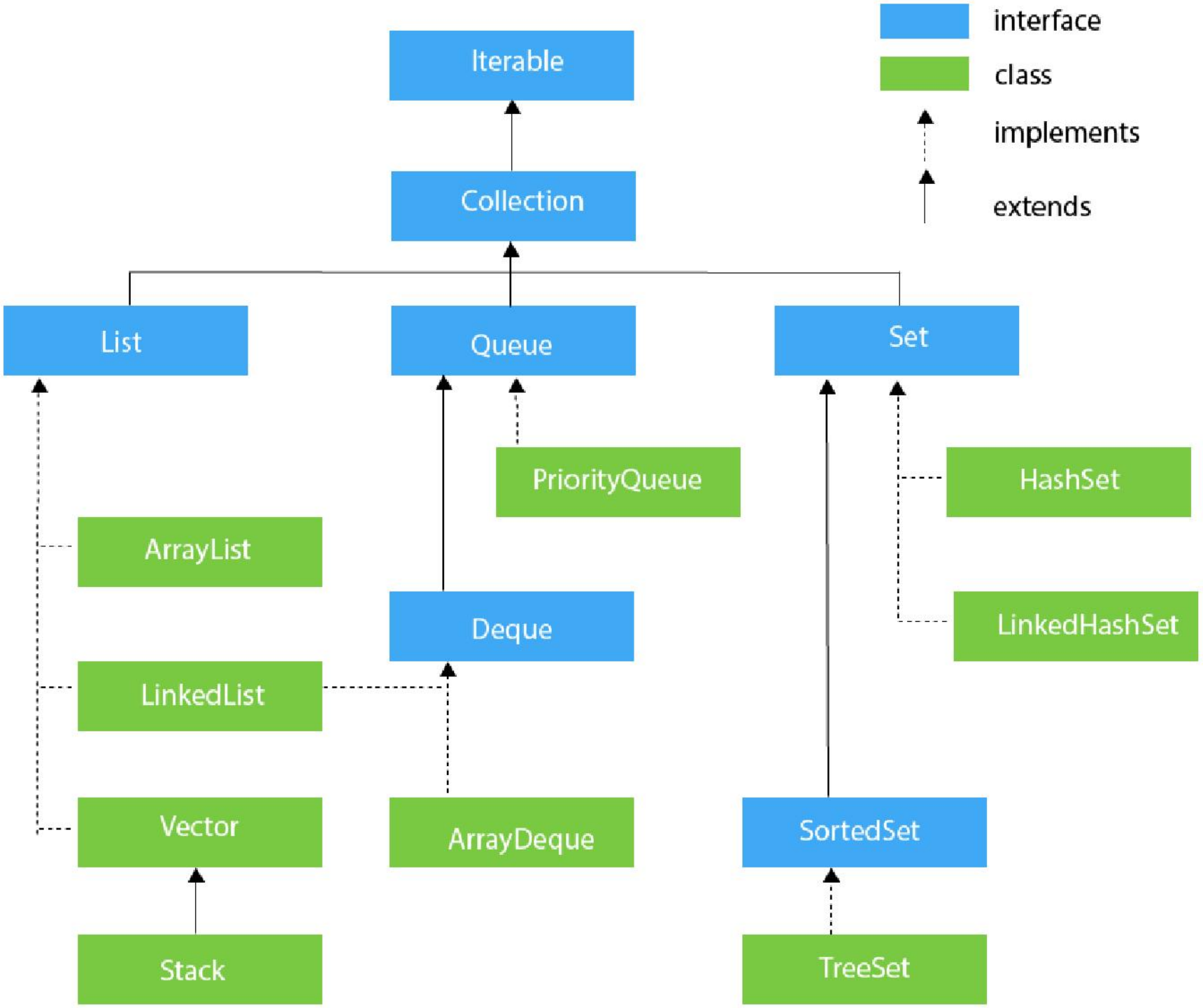
- Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- A Collection represents a single unit of objects, i.e., a group.

Framework :

- It provides readymade architecture.
- It represents a set of classes and interfaces

It has:

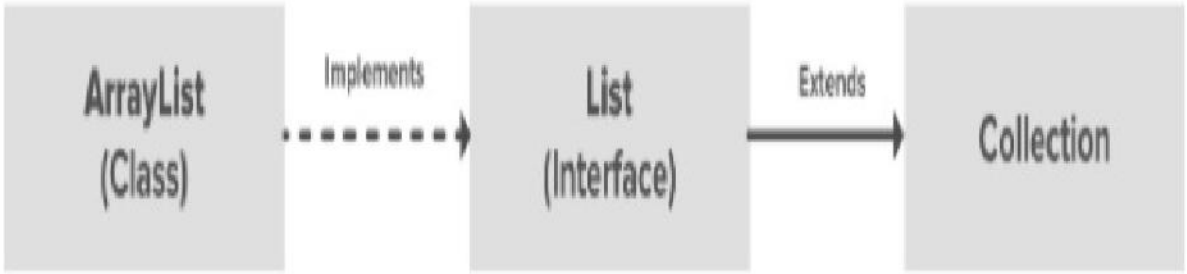
- Interfaces and its implementations, i.e., classes
- Algorithm


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

List

- List in Java provides the facility to maintain the ordered collection.
- It contains the index-based methods to insert, update, delete and search the elements.
- It can have the duplicate elements also. We can also store the null elements in the

The List interface is found in the java.util package



List-Operations

Operation 1: Adding elements to List class using add() method

Operation 2: Updating elements in List class using set() method

Operation 3: Removing elements using remove() method

ArrayList

- The ArrayList class is a resizable array, which can be found in the java.util package.
- While elements can be added and removed from an ArrayList whenever you want.
- ArrayList in Java can have the duplicate elements also.
- It implements the List interface so we can use all the methods of List interface here.
- It maintains the insertion order internally

Demo Code

```
import java.util.*;

public class Main
{
	public static void main(String[] args) {

		int[] arr={12,33,44,02,8};

		//Iterate an array

		for(int i=0;i<arr.length;i++)
		{
		   System.out.println("Integer Array:"+arr[i]);
		}


		//for each(type var:array)

		for(int  x :arr)
		{
		System.out.println(x);
		}
```

```
char[] ch=new char[5];
        ch[0]='b';
        ch[1]='c';
        ch[2]='e';
        ch[3]='a';
        ch[4]='z';

    for(char c:ch)
    {
       System.out.println("Character Array:"+c);
    }


    System.out.println("Accessing based on index:"+ch[2]);


    String[] str={"Java","Python","C#"};

    for(int i=0;i<str.length;i++)
    {
       System.out.println("String Array:"+str[i]);
    }
```

```
String str1="Tata";

        //convert string into char array

        char[] char1=str1.toCharArray();

        for(char c:char1)
        {
           System.out.println("Coverting String into Character Array:"+c);
        }


           int[] arr1={7,8,3,1};

           Arrays.sort(arr1);

           System.out.println("Sorting with array:"+Arrays.toString(arr1));
```

```
//ArrayList
        //Non generic-It is not specific to datatype

        ArrayList list=new ArrayList();

        list.add(10);
        list.add("Java");
        list.add(true);
        list.add(null);
        list.add('c');
        list.add(56.78);

        //iterate an ArrayList


        for(var l:list)
        {

           System.out.println("Non generic :"+l);


        }
```

```
System.out.println("size of arraylist:"+list.size());

        System.out.println("Checking the arraylist:"+list.isEmpty());

        ArrayList list1=new ArrayList();

        System.out.println("size of arraylist:"+list1.size());

        System.out.println("Checking the arraylist:"+list1.isEmpty());

        //Delete an element

       // list.remove('c');
        list.remove(4);

        for(var l :list)
        {

           System.out.println("After removed element:"+l);
        }
```

```
//Access the elements

        System.out.println(list.get(1));

        //update the elements

        list.set(3,"Apple");

        for(var l :list)
        {

           System.out.println("After updation:"+l);
        }



        //generic

        ArrayList<Integer> intlist=new ArrayList<Integer>();
        intlist.add(0,90);
        intlist.add(56);
        intlist.add(23);
        intlist.add(78);
        intlist.add(65);
        intlist.add(67);
        intlist.add(3,45);
```

```
//iterate an ArrayList

        for(int i:intlist)
        {
           System.out.println("Generic:"+i);
        }


        //Sorting

        Collections.sort(intlist);

        for(int i:intlist)
        {
           System.out.println("Ascending Sorting :"+i);
        }



       Collections.reverse(intlist);


         for(int i:intlist)
        {
           System.out.println("Descending order :"+i);
        }
```

```
        ArrayList<String> stlist=new ArrayList<String>();

                stlist.add("Scooter");
                stlist.add("Lorry");
                stlist.add(2,"Cycle");
                stlist.add(3,"Truck");

                //Iterate an ArrayList
                Iterator itr=stlist.iterator();
                while(itr.hasNext())
                {
                   System.out.println("ArrayList String :"+itr.next());//printing the elements
                }


        }
}
```

Assignments

1.Write a Java Program to display the array elements in different ways(for and foreach)
2.Write a Java Program to find the average of array elements
3.Write a Java Program to find the sum of two arrays (array 1+array2)
4.Write a Java Program to compare two arrays
5.Write a Java Program to sort an array in ascending order
6.Write a Java Program to find the number of elements present in the ArrayList
7.Write a Java Program to find whether the given ArrayList is empty or not?
8.Write a Java Program to get an element from a particular position of an ArrayList?
9.Write a Java Program to replace a particular element in an ArrayList with the given element?
10.Write a Java Program to remove an element from a particular position of an ArrayList?
11.Write a Java Program to sort an ArrayList in descending order

Logic Building with Collections 16th Feb 2-3pm

Arrays Vs Collections

Arrays	**Collections(Ex. ArrayList)**
Fixed Size	Size can be changed
Are of same datatype(homogenous)	Can store elements of different data types
Can hold primitive and object types	Can hold only object types
No built-in methods. Need to write whole logic using for loop.	Provide built-in functions for sorting, searching, retrieving

Sample Programs:

- **Find max in array**

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

     int[] ar = {10,20,11,32,15,63,11,45,22};     //initializing an integer array of 9
elements
      int max=0;
       for(int i=0;i<ar.length;i++)              // iterating over array from index 0 to 8
       {
         if(ar[i]>max)                        //checking the value of index with 'max'
variable
          {
                  max=ar[i];                    // element with bigger value gets stored
in 'max'
          }
       }
       System.out.println(max);                //biggest(max) element from the array is
printed
   }
   }
```

- **Find average in array**

```
int[] ar = {10,20,11,32,15,63,11,45,22};
      double sum=0;
       for(int i=0;i<ar.length;i++)
       {
          sum=sum+ar[i];
       }
       System.out.println(sum/ar.length);  //finding average
```

- **Create subset array of all odd numbers in array**

```
int[] ar = {10,20,11,32,15,63,31,45,22};
     int od=0;

       for(int i=0;i<ar.length;i++)
       {
          if(ar[i]%2!=0)
          {
             od=od+1;
          }
       }

       int[] arodd = new int[od];
       int x =0;

       for(int j=0;j<ar.length;j++)
       {
          if(ar[j]%2!=0)
          {
             arodd[x] = ar[j];
             x++;
          }
       }

       for(int k=0;k<arodd.length;k++)
       {
       System.out.println(arodd[k]);
       }
```

- **Swap List from middle** (E.g. for list - 33 44 11 22 77 88 99 55, answer will be 77 88 99 55 33 44 11 22 as we swap first 4 and last 4. If list has odd item count then middle value remains at same position)

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


ArrayList<Integer> ar = new ArrayList<>();
    ar.add(10);
    ar.add(29);
    ar.add(32);
    ar.add(15);
    ar.add(31);
    ar.add(47);
    ar.add(91);

    int s=ar.size();
    int ele = s/2;
    int mod = s%2;

    for(int i=0;i<ele;i++)
    {
       if(mod==0)
    Collections.swap(ar,i,i+ele);
     else
     Collections.swap(ar,i,i+1+ele);
    }

    for(int j=0;j<ar.size();j++)
    {
       System.out.println(ar.get(j));
    }
}
}
```

- **Find count of prime numbers in the list**

```
ArrayList<Integer> ar = new ArrayList<>();
    ar.add(10);
    ar.add(29);
    ar.add(32);
    ar.add(15);
```

```
ar.add(31);
ar.add(47);

int ct=0;

 for(int i=0;i<ar.size();i++)            //iterating over list
 {
    int cn=0;
    for(int j=2;j<ar.get(i);j++)          //loop from 2 to number-1 (excluding 1 and
the number itself)
    {
       if(ar.get(i)%j==0)                 //condition to check if the number is divisible by
any number except 1 and same number
       {
          cn++;
          break;                  //exit from the loop as soon as the number gets
divisible by any other number.

       }
    }
    if(cn>0)
    {                              //counting non prime numbers
       ct++;
    }

 }

 System.out.println(ar.size()-ct);        //printing count of prime numbers.
```

TATA CONSULTANCY SERVICES
Experience certainty.

Topic:Advanced Collections

Set

The set interface is present in java.util package and extends the Collection interface is an unordered collection of objects in which duplicate values cannot be stored

The set interface is used to create the mathematical set.

Set Methods:
1.add()
2.addAll()
3.clear()
4.contains()
5.containsAll()
6.isEmpty()
7. remove()

HashSet

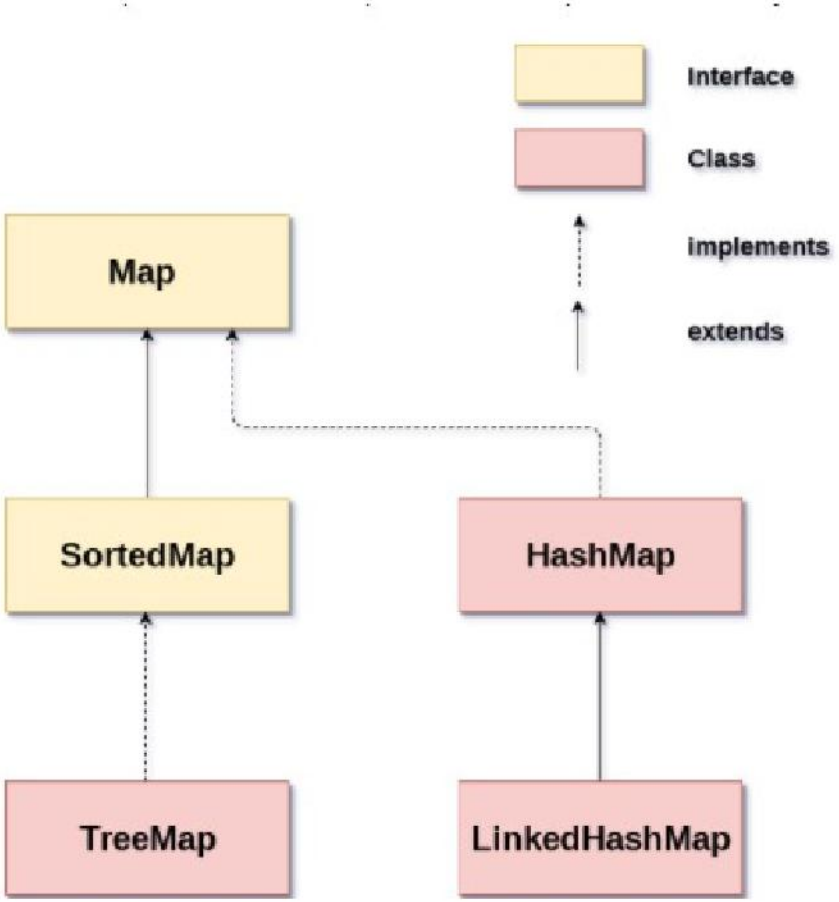
- HashSet class is used to create a collection that uses a hash table for storage.
- HashSet contains unique elements only.
- HashSet allows null value.
- HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.
-
- HashSet does not maintains the order of its elements. Hence sorting of HashSet is not possible
- HashSet can be sorted indirectly by converting into List or TreeSet

TreeSet

- TreeSet class implements the Set interface that uses a tree for storage
- TreeSet class contains unique elements only like HashSet.
- TreeSet class doesn't allow null element.
- TreeSet class maintains ascending order
- TreeSet class access and retrieval times are quiet fast

Map

A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.



Map

A Map doesn't allow duplicate keys, but you can have duplicate values

Map.Entry Interface:

- Entry is the subinterface of Map. So we will be accessed it by Map.Entry name.

- It returns a collection-view of the map, whose elements are of this class. It provides methods to get key and value

HashMap

- HashMap class implements the Map interface which allows us to store key and value pair, where keys should be unique
- If you try to insert the duplicate key, it will replace the element of the corresponding key
- It is easy to perform operations using the key index like updation, deletion etc.
- HashMap may have one null key and multiple null values.
- HashMap maintains no order

TreeMap

- TreeMap class is a red-black tree based implementation. It provides an efficient means of storing key-value pairs in sorted order
- TreeMap contains only unique elements
- TreeMap cannot have a null key but can have multiple null values
- TreeMap maintains ascending order.

Demo Code

```
import java.util.*;

public class Main
{
	public static void main(String[] args) {

	//HashSet - No Duplicate values ,accepts one null value,No insertion order
	//HashSet using Hashing alogirthm
	HashSet<Integer> set=new HashSet<Integer>();
		set.add(90);
		set.add(80);
		set.add(190);
		set.add(78);
		set.add(null);
		set.add(null);
		set.add(78);

	System.out.println("HashSet:"+set);

	//Iterator-Looping the collection

	Iterator itr=set.iterator();
	while(itr.hasNext())
	{
	   System.out.println("HashSet using Iterator:"+itr.next());
	}
```

```
set.remove(80);

 System.out.println("After removed element :"+set);


 System.out.println("Element exists or not :"+set.contains(78));
 System.out.println("Size:"+set.size());
 System.out.println("Set is Empty or not:"+set.isEmpty());

 set.clear();

 HashSet<String> set1=new HashSet<String>();
 set1.add("Java");
 set1.add("Python");

 HashSet<String> set2=new HashSet<String>();
 set2.add("C#");
 set2.add("HTML");
 set1.addAll(set2);

 System.out.println("Set1 values:"+set1);
 System.out.println("Set2 values:"+set2);
```

```
//TreeSet -No Duplicate values,Does not allow null values, maintains ascending order

	TreeSet<Character> tset=new TreeSet<Character>();
	tset.add('C');
	tset.add('A');
	tset.add('Y');
	tset.add('B');
	tset.add('B'); // no duplicate values
//	tset.add(null);

	System.out.println("Tree Set :"+tset);

	System.out.println("Tree Set Descending order :"+tset.descendingSet());

	//tset.add(0,'H'); //cannot insert based on index
	//tset.remove(1);//cannot remove based on index
	tset.remove('A');
	System.out.println("After removed element :"+tset);

	TreeSet<Integer> tset1=new TreeSet<Integer>();
	tset1.add(100);
	tset1.add(101);
	tset1.add(102);
	tset1.add(103);
```

```
        System.out.println("Tree set :"+tset1);

        System.out.println("Head Set:"+tset1.headSet(102));

        System.out.println("Tail Set:"+tset1.tailSet(102));

        System.out.println("Sub Set:"+tset1.subSet(100,102));

        System.out.println("Highest Value:"+tset1.pollFirst());

        System.out.println("Lowest Value:"+tset1.pollLast());

      //MAP

        //HashMap-Key value pair, Duplicate key will replace the corrsponding
key,No insertion Order
        //accepts multiple null values,accept one null key

          HashMap<Integer,String> map=new HashMap<Integer,String>();
              map.put(8,"Aman");
              map.put(2,"Aman");
              map.put(9,"Sandy");
              map.put(4,"Zyan");
              map.put(5,null);
              map.put(6,null);
              map.put(9,"Preeti");//duplicate key replaced the existing value
              map.put(null,"Nasrath");
              //map.put(null,"Hismath");
```

```
System.out.println("HashMap:"+map);

        for(Map.Entry m:map.entrySet())
        {
           System.out.println("HashMap using Map Entry Interface:"+m.getKey()+"
"+m.getValue());
        }

        System.out.println(map.get(4));
        System.out.println(map.remove(2));

        System.out.println("After removed the element:"+map);


        //TreeMap-Maintains ascending order,Cannot have null key
        //accepts multiple null values



        TreeMap<Integer,String> tmap=new TreeMap<Integer,String>();
        tmap.put(1,"Aman");
        tmap.put(5,"Aman");
        tmap.put(3,"Sandy");
        tmap.put(4,"Zyan");
        tmap.put(6,null);
        tmap.put(2,null);
        tmap.put(2,"Preeti");//duplicate key replace the existing the value
        //tmap.put(null,"Nasrath");//cannot accept null key
```

```
  System.out.println("Tree Map:"+tmap);

  System.out.println("Tree Map in Descending Order:"+tmap.descendingMap());

  //Map Entry

for(Map.Entry m:tmap.entrySet())
      {
        System.out.println("TreeMap using Map Entry Interface:"+m.getKey()+" "+m.getValue());
      }


      for(int i : tmap.keySet())
      {
        System.out.println("Keys :"+i);
      }

   for(String s:tmap.values())
   {
      System.out.println("Values :"+s);
   }
```

```
    TreeMap<Integer,String> map1=new TreeMap<Integer,String>();
     map1.put(100,"Amit");
     map1.put(101,"Ravi");
     map1.put(102,"Vijay");
     map1.put(103,"Rahul");



       System.out.println("Tree Map :"+map1);

       System.out.println("Head Map:"+map1.headMap(102));

       System.out.println("Tail Map:"+map1.tailMap(102));

       System.out.println("Sub Map:"+map1.subMap(100,102));




       }
}
```

Assignments

1. Write a Java Program to count the number of elements in the HashSet?
2. Write a Java Program to remove the specified element from HashSet?
3. Write a Java Program to return the first element from the TreeSet?
4. Write a Java Program to display the elements in descending order using TreeSet?
5. Write a Java program to find duplicate characters in a string using hashmap?
6. Write a Java Program to Find the Occurrence of Words in a String using HashMap?

Java Basics

Searching and Sorting

Topics

- Iterating(looping) an Array/ArrayList using a for or for-each loop(Recap)
- Searching an Array
- Searching a List (ArrayList)
- Sorting an Array
- Test Your Knowledge

Iterating(looping) an Array using a for loop

```
int[ ]  nums = { 2,4,6,8,10};

for ( int i=0 ;i<nums.length; i++){

    System.out.println(nums[i]);

}
```

iteration	Value of loop counter variable(i)	nums[i]	Value
1	0	nums[0]	2
2	1	nums[1]	4
3	2	nums[2]	6
4	3	nums[3]	8
5	4	nums[4]	10

What is the value of nums.length ?
What is the final value of i?

Iterating(looping) an Array using a for-each loop

```
int[ ]  nums = { 2,4,6,8,10};

for ( int x : nums ) {

   System.out.println(x);

}
```

iteration	x
1	2
2	4
3	6
4	8
5	10

In for-each loop, we need to declare a variable that is the same type as the data-type of the array instead of declaring and initializing a loop counter variable.

Iterating(looping) an ArrayList using a for loop

```
ArrayList list = new ArrayList();
list.add(2);
list.add(4);
list.add(6);
list.add(8);
list.add(10);

for(int i=0 ; i<list.size();i++){

    System.out.println(list.get(i));

}
```

iteration	Value of loop counter variable(i)	list.get(i)	Value
1	0	list.get(0)	2
2	1	list.get(1)	4
3	2	list.get(2)	6
4	3	list.get(3)	8
5	4	list.get(4)	10

- list.size() in ArrayList and array.length in Array.
- list.get(i) in ArrayList and array[i] in Array.

Iterating(looping) an ArrayList using a for-each loop

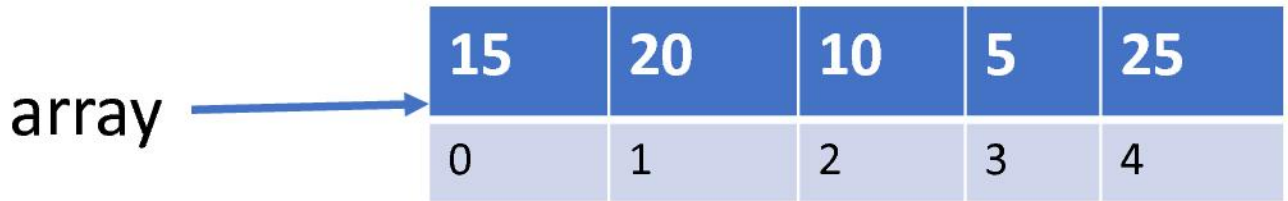
```
ArrayList<String> list = new ArrayList<String>();
list.add("Orange");
list.add("Mango");
list.add("Apple");

for(String fruit : list){
    System.out.println(fruit);
}
```

iteration	fruit
1	Orange
2	Mango
3	Apple

Searching an Array

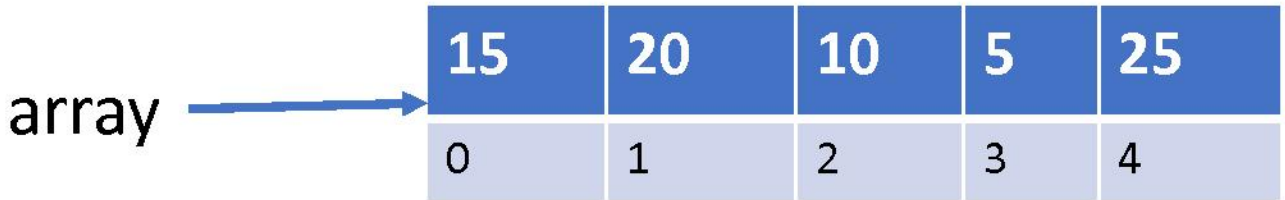
- For searching an array , simply iterate the array using for/for-each loop and in each iteration check if the current element matches the condition.



Search if the number 10 exists in the array.

```
for(int i = 0; i<array.length; i++){
  if (array[ i ] == 10 ){ //logic }
}
```

Searching and Array(cont..)



Search an array and check if the value in a variable x available in the array.

```
int x = 22;
for(int i = 0; i<array.length; i++){
  if (array[ i ] == x ){
    System.out.println("Number available.");
      break;
  }
}
```

Searching and ArrayList

- Searching an ArrayList is similar to searching an array.

```
ArrayList<Integer> list = new ArrayList();
list.add(2);
list.add(4);
list.add(6);
list.add(8);
list.add(10);

for(int i=0 ; i<list.size();i++){
    if(list.get(i) > 5)
    System.out.println(list.get(i));

}
```

Problem 1

- Create an array of n numbers.
- Read two numbers(range) as input and check how many numbers in that range exists in the array.
- Print the count if numbers in that range exists in the array else print “No number in this range exists”.

Problem1 : Solution

```
public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int[] nums = { 22,33,44,55,66,77,88,99 };
        System.out.println("Enter two numbers");
        int x = scan.nextInt();
        int y = scan.nextInt();
        int c=0; // variable to store the count
        for( int i=0 ; i<nums.length; i++){
            if(nums[i] >= x && nums[i]<=y){
                c++;
            }//end if
        }//end for

        if(c > 0)
        System.out.println( c+" numbers available in this range.");
        else
        System.out.println( "No numbers available in this range.");
}
```

Problem 2

- Create an ArrayList and add names of five fruits into it.
- Read a fruit name as an input and check if it is available in the ArrayList.
- If available then print “Fruit Available” else print “Fruit Not Available”.

Problem2 : Solution

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
      //create arraylist and add 5 fruit names
    ArrayList<String> fruits = new ArrayList<String>();
    fruits.add("Mango"); fruits.add("Papaya");
    fruits.add("Orange");fruits.add("Apple");fruits.add("Grapes");
    System.out.println("Enter Fruit Name");
    String fruit = s.nextLine(); //read fruit name
    /*initialize fruitAvailable variable
     - indicates if fruit name is available or not. */
    boolean fruitAvailable = false;  // assume fruit is not available
    for(int i=0;i<fruits.size();i++ ){
        // if fruit available change the value of fruitAvailable variable into true
        if(fruits.get(i).equals(fruit)){ //use equals() method instead of == operator
            fruitAvailable=true;
            break; // stop and exit from the loop
        }//end if
    }//end for
    if(fruitAvailable){
        System.out.println("Fruit Available "+fruit);
    }
    else{
        System.out.println("Fruit Not Available ");
    }
}
```

Problem 3

- Search and find the smallest number in an array of n numbers.

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

    int[] nums = { 13, 4, 14, 7,3, 20};

    int smallest = nums[0];//assume that the first number in the array is the smallest number
    //iterate the array
    for(int i=0; i<nums.length;i++){
        //check if any number in the array is smaller than the current smallest number
        if(nums[i] < smallest){
            //change the smallest number
            smallest = nums[i];
        }
    }

    System.out.println("Smalest Number is "+smallest);

}
```

Problem 4

- Swap the numbers in the first and last position in an array.

Array before swapping →

22	11	55	33	44

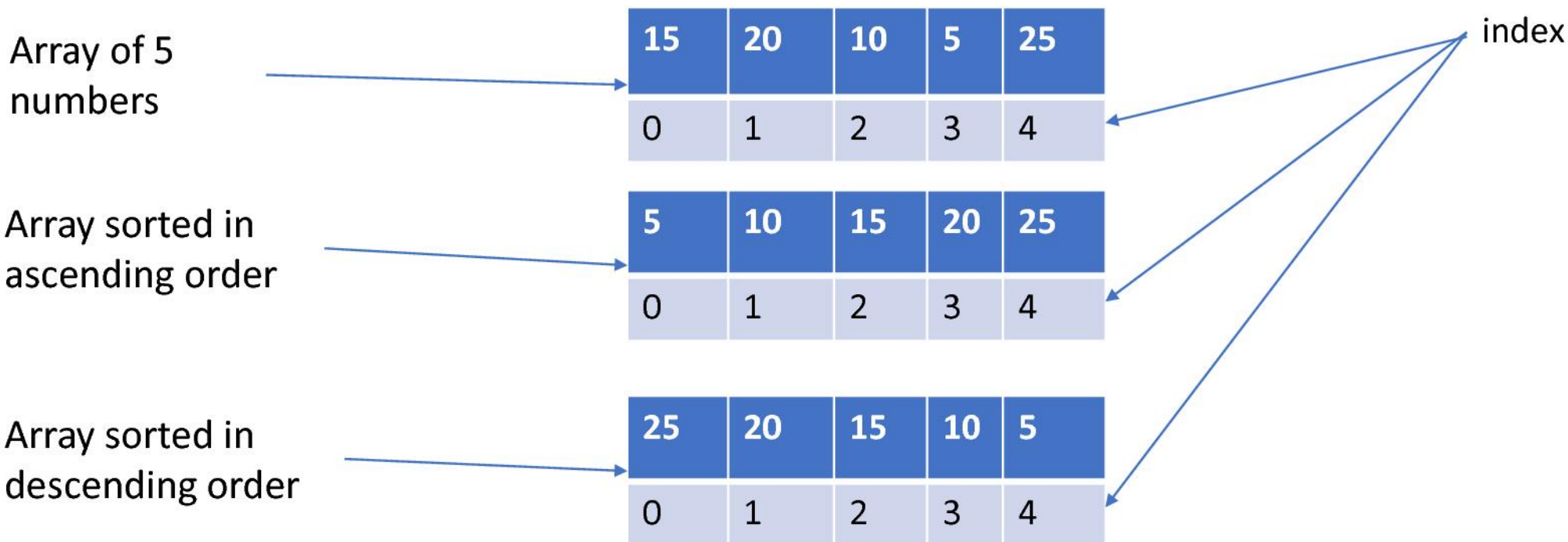
Array after swapping →

44	11	55	33	22

```
int temp = array[0];
array[0] = array[array.length-1]; //array[3]
array[array.length-1] = temp;
```

Sorting an Array

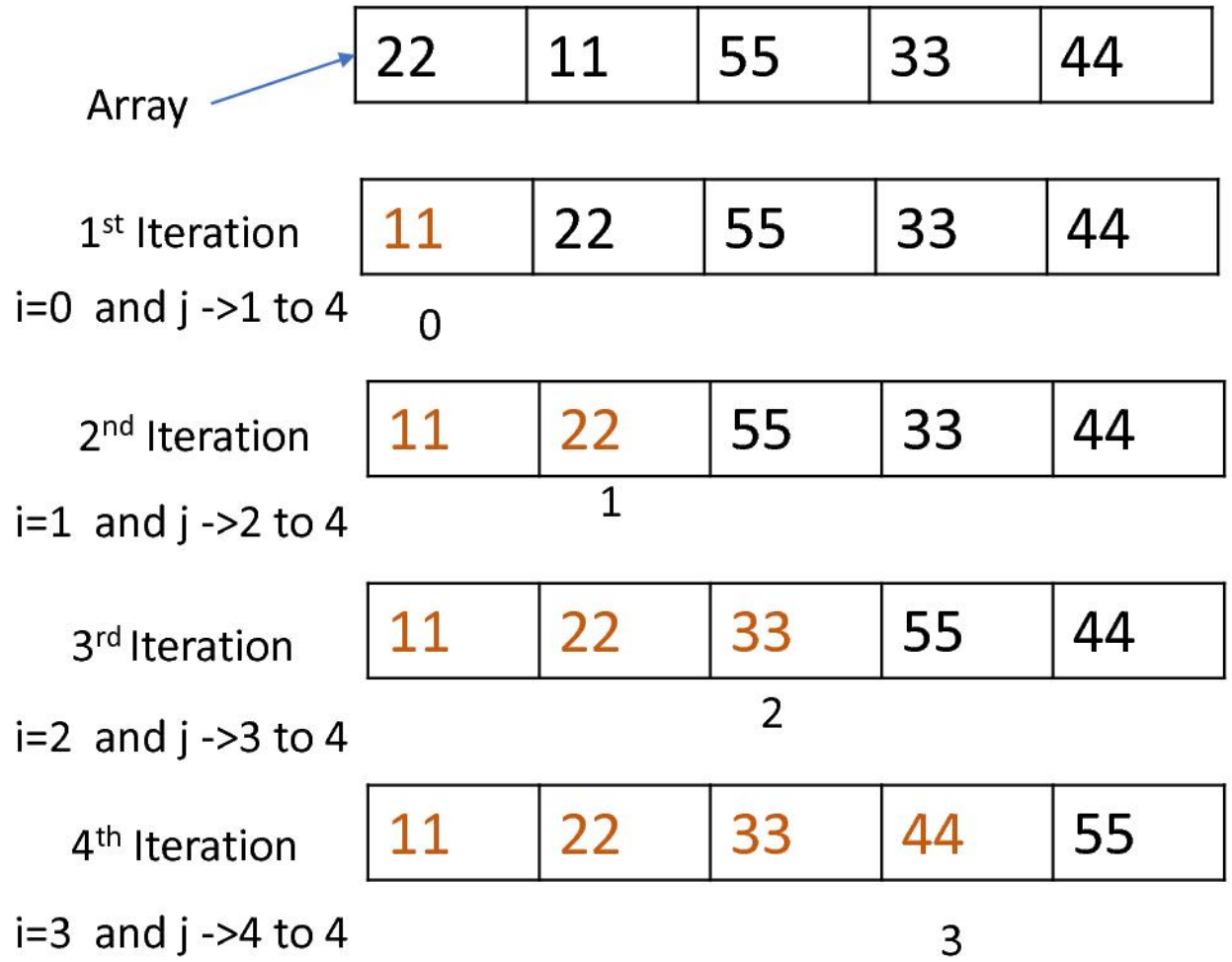
- Sorting and array means rearranging the elements in the array in a specific order.



Sorting an Array (Ascending Order)

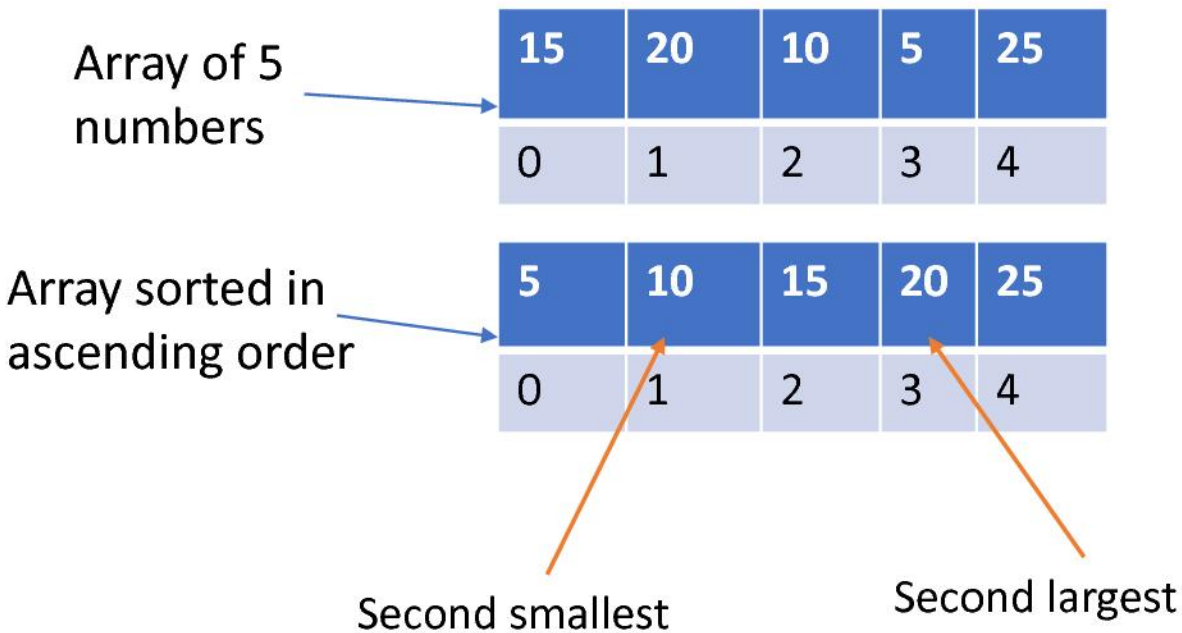
```
FOR i = 0 to array.length-1
    FOR j = i+1 to array.length-1
        IF array[j] < array[i]
            SWAP elements in i and j position
```

- *Outer loop variable i indicates the starting position of each search.*
- *Inner loop is used for searching the array and finding the smallest number.*



Problem 5

- Find the second smallest/second largest element in an array of n numbers.



Problem 5 - Solution

```
public static void main(String[] args)
{   int size, temp;
    Scanner scan = new Scanner(System.in);
    System.out.print("Enter the size of the array: ");
    size = scan.nextInt();    // read the size of the array
    int nums[] = new int[size]; //creating the array
    System.out.println("Enter array elements:");
    for (int i = 0; i < size; i++) {  nums[i] = scan.nextInt(); }  //reading values into the array
       //start sorting
    for (int i = 0; i < size; i++)
    {    //find the smallest number and put it in the i-th position
        for (int j = i + 1; j < size; j++) {
            if (nums[j] < nums[i])  // swap the numbers in i and j position
            {   temp = nums[i];
                nums[i] = nums[j];
                nums[j] = temp;
            }
        }//end inner for
    }//end outer for
    System.out.println("Second Smallest Number "+nums[1]);
    System.out.println("Secons Largest Number"+nums[size-2]);
}
```

```
Enter the size of the array: 5
Enter array elements:
4
5
3
2
1
Second Smallest Number : 2
Secons Largest Number : 4
```

Test Your Knowledge 1

- What is the output of the below program?

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

int[] nums =  { 11,22,33,44,55,66,77,88,99};

   for(int i=0;i<nums.length;i++){
         if(nums[i]%2==0)
             System.out.println(nums[i]);
   }

}
```

Ans: This program will print all even numbers in the array.
22
44
66
88

Test Your Knowledge 2

- What is the output of the below program.

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

    String[] names = { "James","John","Raj","Rohit","Aamir"};

    for( String name : names){
        if(name.contains("a")){
            System.out.println(name);
        }
    }

}
```

Ans: This program will print all the names that contain the character ‘a’ .

James
Raj
Aamir

Test your Knowledge-3

- What is the output of the below program?

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

int[] nums =  { 11,22,33,44,55,66,77,88,99};

   for(int i=0;i<nums.length;i++){
         if(nums[i]%2==0)continue;
             System.out.println(nums[i]);
   }

}
```

Ans: This program will print all the odd numbers in the array .
11
33
55
77
99

Note: Continue is an instruction to skip the current iteration and continue with the next iteration.

Thank You!

Class, Attributes, Constructor, Methods, Objects – Java

23-Feb	2 pm to 3 pm

An architect will have the blueprints for a house. Those blueprints will be plans that explains exactly what properties the house will have and how they are all laid out. However, it is just the blueprint, you can't live in it. Builders will look at the blueprints and use those blueprints to make a physical house. They can use the same blueprint to make as many houses as they want. **Each house will have the same layout and properties** and can accommodate it's own family.

Here the blueprint can be considered as Class and each house as the Object created from the class.

Class = attributes/fields + Methods

Fields of a class are known as **instance** variables.(outside any method)

Fields within a method are called Local variables.

Methods are used to perform some action using the fields/attributes.

```
returnType MethodName (parameter-list)
{
    Method-body;
}
```

```
public class Employee{

   int empid;       //instance variables

   String empname;

   double salary;

   double updateSalary()
```

```
    {
        return salary+5000;
    }
}
```

Object : any physical or logical entity having some characteristic or can perform some work.
Eg. A pen or college management system

New datatype-> Employee is created in previous example.

To declare an object we can use the same way as we declare any other datatype(int/String etc)

```
int a;
Employee e;     //declaring an object
```

Objects are created using NEW keyword followed by call to constructor.

```
Employee e1 = new Employee();
Employee e2 = new Employee();
```

How do we access data of Employee class using objects.

```
Objectname.variablename/methodname
e1.empid
e2.empname
e1.updateSalary()
```

```
double newsal = e1.updateSalary()
```

Each time a new object is created, at least one constructor will be invoked.

Default or parametrized.

```
class Employee{
   int empid;       //instance variables
   String empname;
   double salary;

   double updateSalary()
   {
      return salary+5000;
   }
}

public class MyClass{
   public static void main(String[] args)
   {
      Employee e1 = new Employee();   //calling constructor
      System.out.println(e1.empid);
```

```
        double newsal = e1.updateSalary();     //local variable
        System.out.println(newsal);
    }
}
```

With parameters:

```
class Employee{
    int empid;       //instance variables
    String empname;
    double salary;
    Employee(int id,String name,double sal)
    {
        this.empid=id;
        this.empname=name;
        this.salary=sal;
    }
    double updateSalary()
    {
        return salary+5000;
    }
}
```

```
public class MyClass{
   public static void main(String[] args)
   {
      Employee e1 = new Employee(111,"Akash",20000.50);


      Employee e2= new Employee(112,"Raman",17000);


      System.out.println(e1.empid);
      double newsal = e1.updateSalary();
      System.out.println(newsal);
      System.out.println(e2.empid);
      double newsal2 = e2.updateSalary();
      System.out.println(newsal2);
   }
}
```

Comparison of objects

```
public class MyClass{
   public static void main(String[] args)
   {
      Employee e1 = new Employee(111,"Akash",20000.50);
```

```
    Employee e2= new Employee(112,"Raman",17000);


    //salary comparison
        if(e1.salary > e2.salary)
        System.out.println("1st employee has more salary");
    else
            System.out.println("2nd employee has more salary");
  }
}
```

Calling Method by passing object

```
public class MyClass{
  public static void main(String[] args)
  {
    Employee e1 = new Employee(111,"Akash",20000.50);
    Employee e2= new Employee(112,"Raman",17000);
    //calling method to print salary of an object
    method1(e1);
  }
    public static void method1(Employee e)        //method definition to
print salary of an object
    {
```

```
        System.out.println("salary of employee is "+e.salary);
    }}
```

Method overloading

declaring multiple methods with same name but different parameters in the same class.

Below example has 2 methods with same name but different datatype in parameters.

```
public class MyClass
{
 void sum (int a, int b)
 {
  System.out.println("sum is "+(a+b)) ;
 }
 void sum (float a, float b)
 {
  System.out.println("sum is "+(a+b));
 }
 public static void main (String[] args)
 {
  MyClass  cl = new MyClass();
```

```
   cl.sum (8,5);
   cl.sum (4.6f, 3.8f);
  }
}
```

Below example has 2 methods with same name but different number of arguments/params

```
public class MyClass
{
 void sum (int a, int b)
 {
  System.out.println("sum is "+(a+b)) ;
 }
 void sum (int a, int b, int c)
 {
  System.out.println("sum is "+(a+b+c));
 }
 public static void main (String[] args)
 {
  MyClass  cl = new MyClass();
  cl.sum (8,5);
```

```
    cl.sum (4,3,7);
  }
}
```

Java Basics

Array of Objects

Topics

- Class and Objects
- Array of Objects
- Array of Object as a method parameter
- Array of Object as a method return type

Class / Properties / Constructors

```
class Book {      } -- class skeleton

class Book {  int id ;  String title ; double price; }  -- adding properties

class Book {     --adding constructors
        int id ;  String title ; double price;
        Book(){  }  -- constructor1
        Book(int id}  { this.id = id;} – constructor2
        Book(int id , String title , double price}  {-- constructor3
                this.id = id;
                this.title=title;
                this.price=price;
            }
        }
```

Creating Objects / Reference Variables

- Use new operator

```
new Book();   -- using constructor1
new Book(100); -- using constructor2
new Book(200, “Java” ,2000.0); -- using constructor3
```

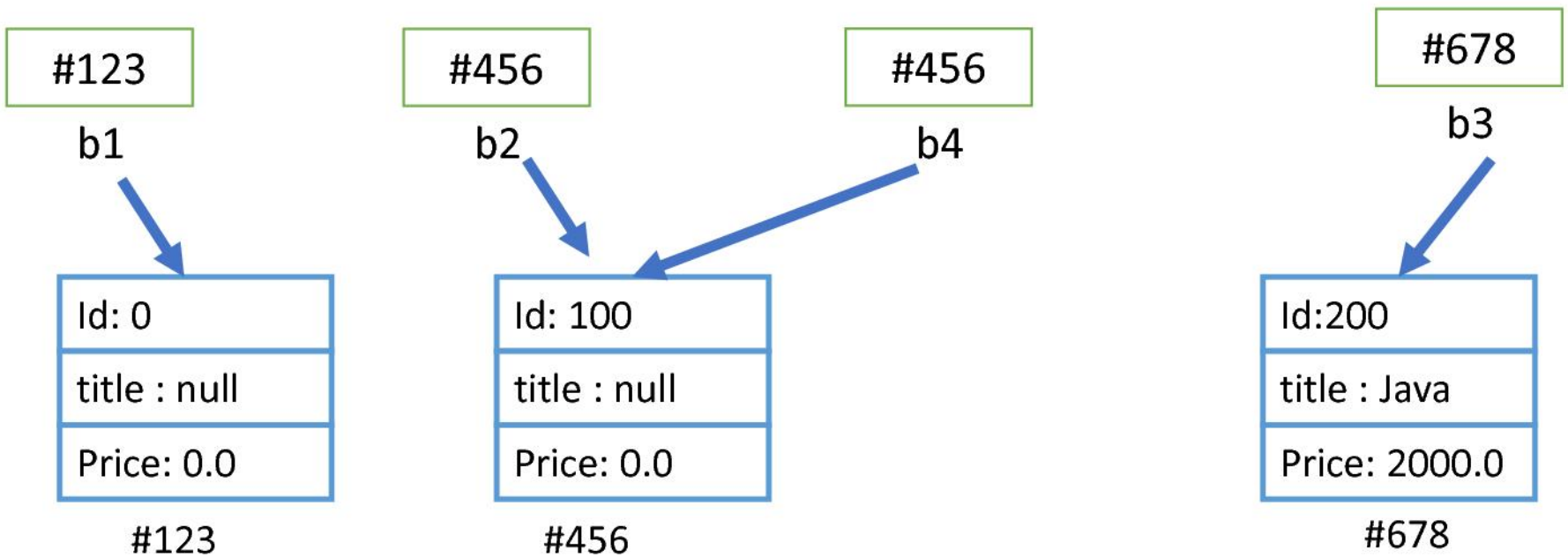
- Store the reference(address) of an object in a reference variable of its type

```
Book b1 = new Book();
Book b2 = new Book(100);
Book b3 = new Book(200, “Java” ,2000.0);
Book b4 = b2
```

Here b1, b2, b3 and b4 are reference variables.

Objects and Reference Variables

```
Book b1 = new Book();
Book b2 = new Book(100);
Book b3 = new Book(200, “Java” ,2000.0);
Book b4 = b2;
```



Methods

```
Return Type  methodName( Parameters ) {
    //body –logic
   [return value ] // need to return a value if return type is not void
 }
```

```
void sayHello( ) {   System.out.println("Hello"); }
```

```
void sayHello( String name) {   System.out.println("Hello"+name); }
```

```
int sum( int x, int y) {   return x+y; }
```

```
Book  findById( Book[] list , int id) {
        for(Book b: list) { if (b.getId()==id) return b;}
          return null;
}
```

Array

Syntax :

datatype[ ] arrayName = { value1, value2,…} ;

E.g. Int[] nums = {11,22,33,44,55};

datatype[ ] arrayName = new datatype[size];

E.g. int[] nums = new int[5];

nums[0] = 11;

nums[1] = 22;

nums[2] = 33;

nums[3] = 44;

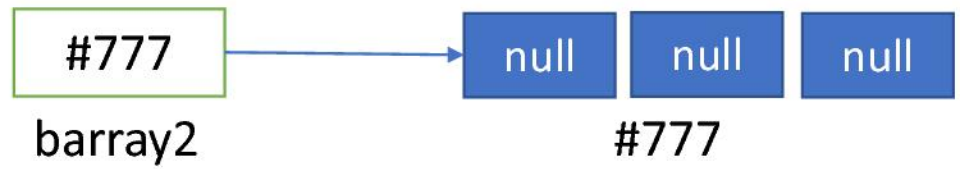
nums[4] = 55;

Array of Objects

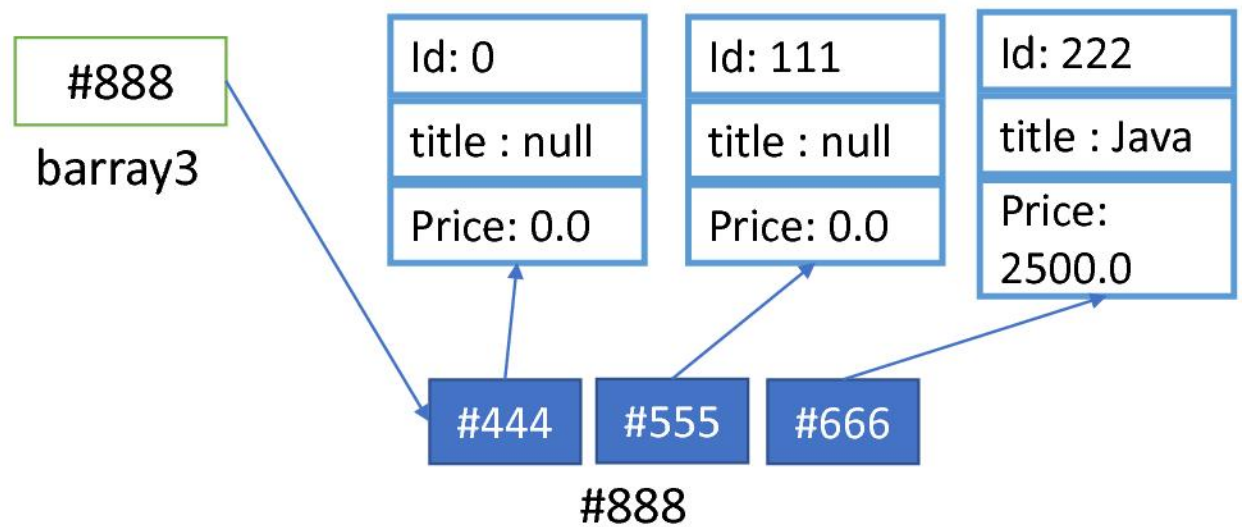
```
class Book {  int id ;  String title ; double price; }
```

```
Book[] barray1 = null;
```


```
Book[] barray2 = new Book[3]
```




```
Book[] barray3 = new Book[3]
 barray3[0] = new Book();
 barray3[1] = new Book(111);
 barray3[2]= new Book(222,"Java", 2500.0);
```



Array as a method parameter and return type

```
Type[ ] MethodName(Type[ ] arrayName ){
  //body
}
```

Type *can be a primitive type or reference type (class or interface)*

Example :

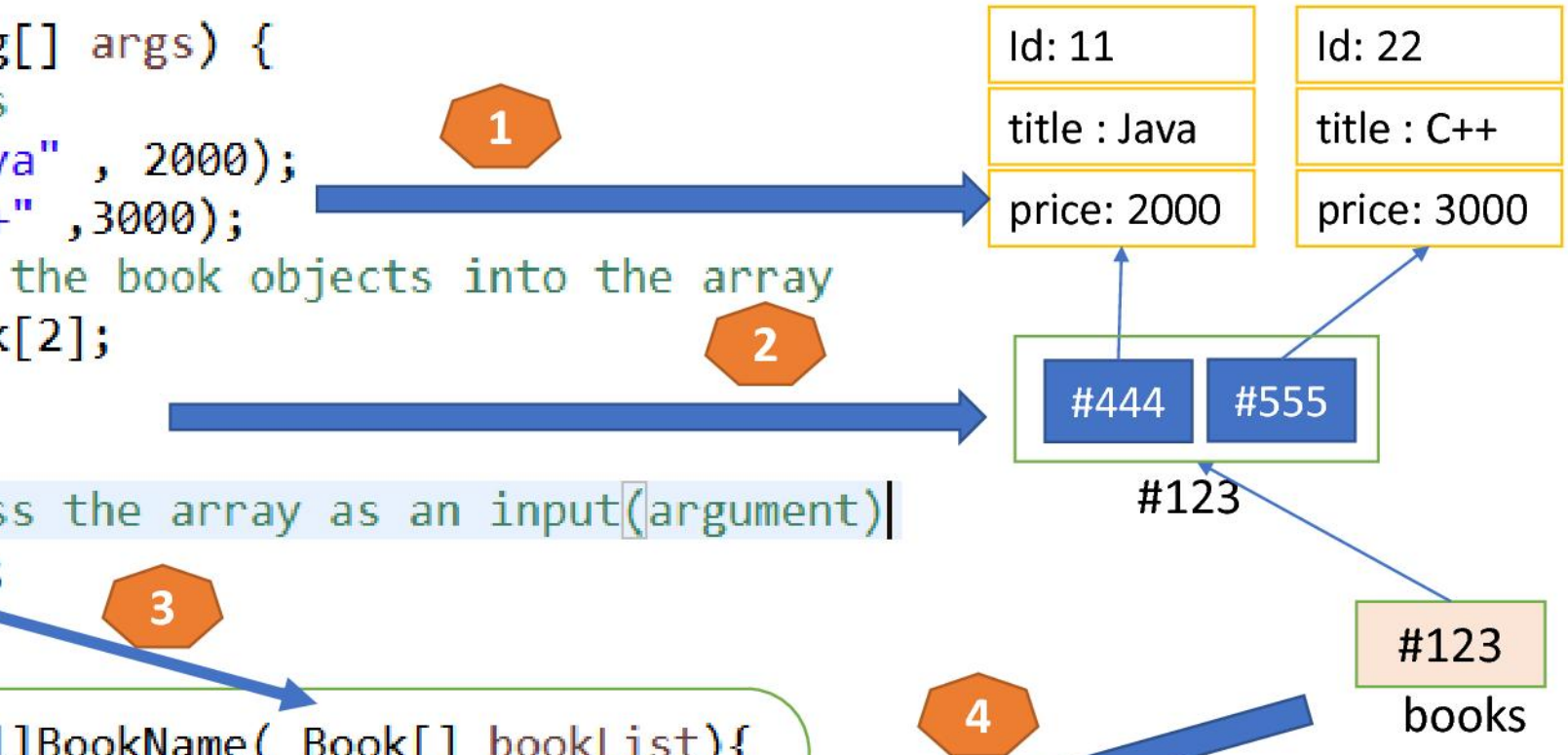
1. int findAverage(int[] nums)
2. String[ ] getSubArray(String[ ] names , int startIndex , int endIndex)
3. void printAllEmployee(Employee[ ] emps) { //body }
4. Employee findEmployeeById(Employee[ ] emps , int id) { //body }
5. Employee[ ] getEmployeeByDept(Employee[ ] emps, String dept)

Example : Passing Array of objects to a method

```
public static void main(String[] args) {
    //crating two book objects
    Book b1 = new Book(11,"Java" , 2000);
    Book b2 = new Book(22,"C++" ,3000);
    //create an array and add the book objects into the array
    Book[ ]  books  = new Book[2];
    books[0] = b1;
    books[1] = b2;
    // call the method and pass the array as an input(argument)
      printAllBookName(books);
    }

    public static void printAllBookName( Book[] bookList){

    for(Book b: bookList){
        System.out.println(b.getTitle());
        }
    }
}
```



Example : Returning an Array of Objects from a method(1)

```
public static void main(String[] args) {
    //call createArray method to create and return the array
     Book[] barray = createArray();
    //print book title
    for(Book b : barray){
        System.out.println(b.getTitle());
    }
}
```

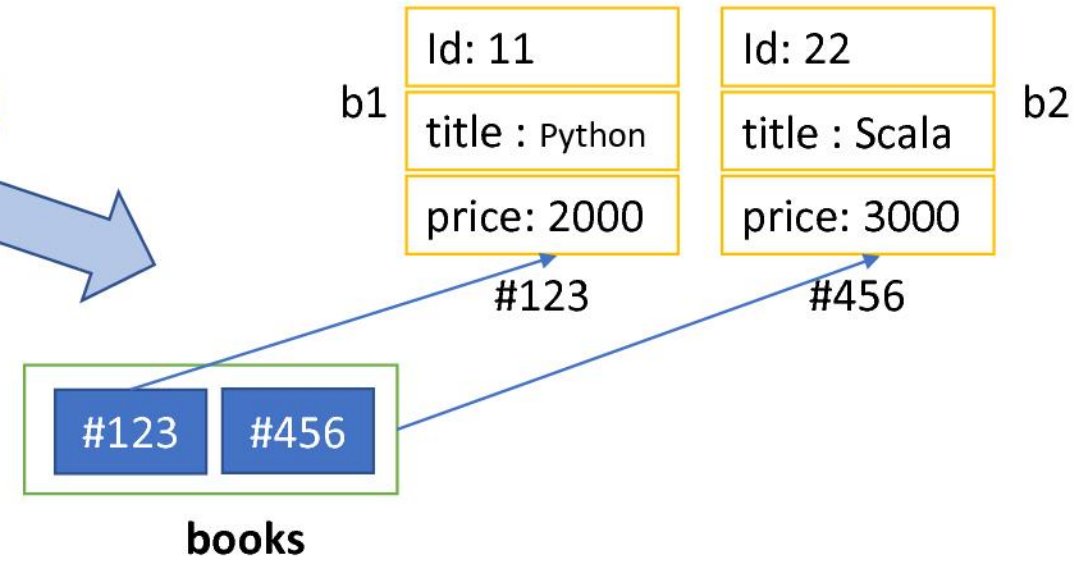
1

```
public static Book[] createArray(){
    Book[] books = new Book[2];
   // create two book objects
    books[0] = new Book(11,"Python" , 2000);
    books[1] = new Book(22,"Scala" ,3000);

    return books;
}
```

2

3



Example : Returning an Array of Objects from a method(2)

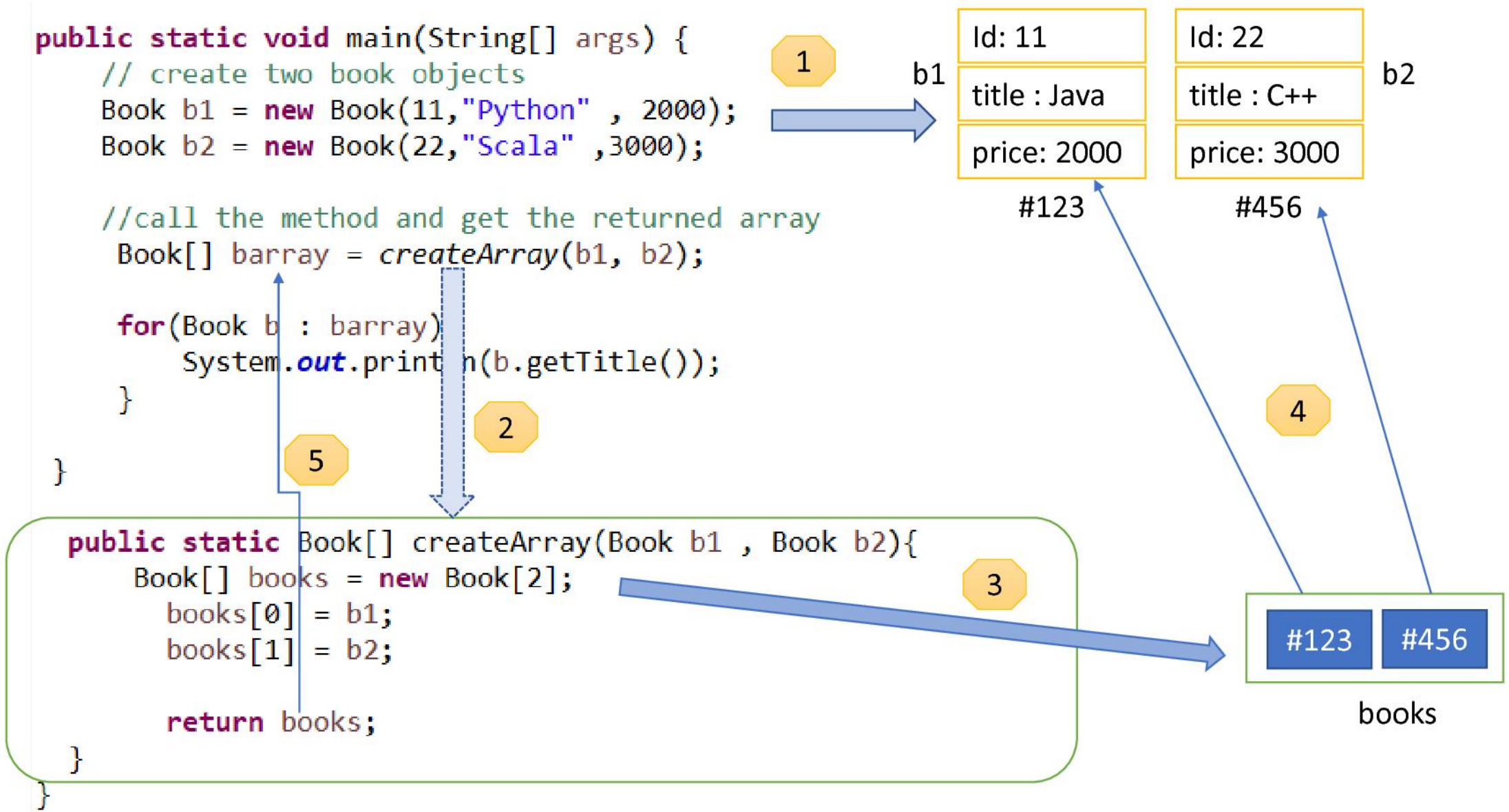
```
public static void main(String[] args) {
    // create two book objects
    Book b1 = new Book(11,"Python" , 2000);
    Book b2 = new Book(22,"Scala" ,3000);

    //call the method and get the returned array
     Book[] barray = createArray(b1, b2);

     for(Book b : barray)
         System.out.println(b.getTitle());
     }
}
```

```
public static Book[] createArray(Book b1 , Book b2){
     Book[] books = new Book[2];
       books[0] = b1;
       books[1] = b2;

       return books;
}
}
```



Problem1

- Create a method that finds the book with the given id from an array of book objects.
  - Method Name : getBookById
  - Method Parameters : Array of book Objects and id of a book
  - Return value : Book object if book with given id available else null value

  - In the main program call the method with proper inputs and check the return value.
  - If the return value is null print “No book available” else print the book details.

Problem1 – Solution (Simple)

```
public static void main(String[] args) {
    Book[] bookArray = new Book[3];
    bookArray[0] = new Book(11,"Java",2000);
    bookArray[1] = new Book(22,"Python",3000);
    bookArray[2] = new Book(11,"Java",2000);
    //call the method
    Book result = getBookById(bookArray, 44);
    if(result !=null){
        System.out.println(result.getTitle());
    }
    else{
        System.out.println("No book available");
    }
}
```

```
public static Book getBookById( Book[] bookList , int id){
    Book book = null;
    for(Book b : bookList){
        if(b.getId()== id){
            book = b;
            break;
        }
    }
    return book;
}
```

Problem1 – Solution (Using Scanner)

```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    //get size of array
    int size  = scan.nextInt();
    //  create book array
    Book[] barray  = new Book[size];
    // add book objects into the array using for loop
    for(int i=0; i<barray.length;i++){
        int id = scan.nextInt(); //read id
        scan.nextLine(); // -- read the newline character
        String title = scan.nextLine();//--read title
        double price = scan.nextDouble();//-- read price
        //create object and add the object into the array
            barray[i] = new Book(id,title,price);
      }//end for
    // read the book id to search in the array
        int searchId = scan.nextInt();
    // call the method
        Book result = getBookById(barray, searchId);
    // print the result
         if(result==null){
             System.out.println("No book available. ");
         }
         else{
             System.out.println(result.getId());
             System.out.println(result.getTitle());
             System.out.println(result.getPrice());
         }
}//end main
```

```
public static Book getBookById( Book[] bookList , int id){
    Book book = null;
    for(Book b : bookList){
        if(b.getId()== id){
            book = b;
            break;
        }
    }
    return book;
}
```

Test Your Knowledge

- What will be the output of below program?

```
public static void main(String[] args) {
    Book b1 = new Book(11,"Java" , 2000);
    Book b2 = new Book(22,"C++" ,3000);
    Book[ ]  books  = new Book[2];
    books[0] = b1;
    books[1] = b2;
    increaseBookPrice(books);
    System.out.println(books[0].getPrice());
    }

    public static void increaseBookPrice( Book[] bookList){
    for(Book b: bookList){
        if(b.getTitle().equals("Java")){
            b.setPrice(b.getPrice()+100);
        }
      } //end for
    }//end method
```

Thank You