Data Handling and Functions

Course Outline

1. Introduction to Java
2. **Data Handling and Functions**
3. Object Oriented Programming in Java
4. Packages and Multithreading
5. Collections

Objectives

After completing this module, you should be able to:

- Implement Single and Multi dimensional array
- Declare and Define Functions
- Call Functions by value and by reference
- Implement Method Overloading
- Use String data-type and String-buffer

Introduction to Arrays

Meet John Again

Remember John? Well, he is now grown up as a Programmer. He has managed to learn Java. Lets see how he is utilizing it.

One day in office..

Hello John and Daisy. You will be glad to know that we have received a very interesting project. And I want you two to work on it.

One day in office..

I will provide the document containing the details. Lets come up with a good solution.

John’s Project – Question Statement

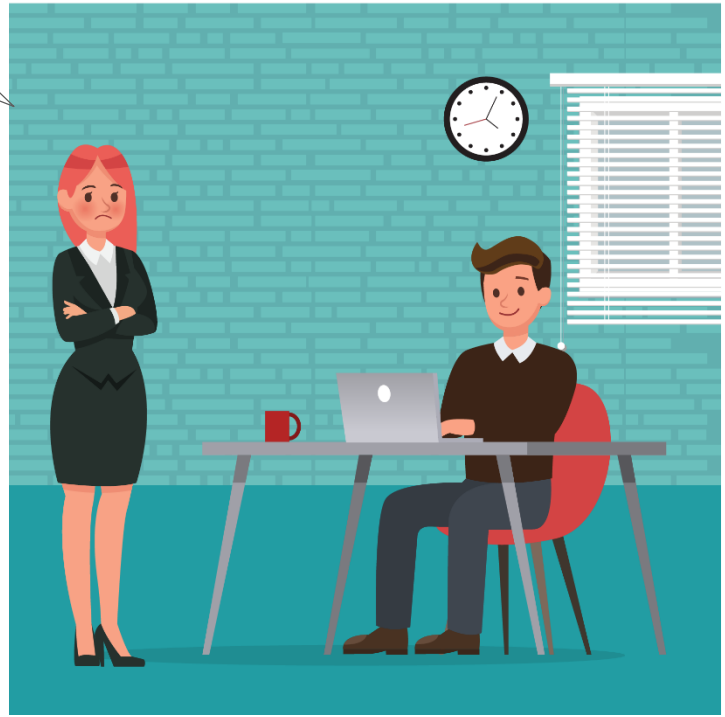
Given below is the data on the votes for different parties, collected during elections in US

Serial no.	Region	Democratic Party	Republican Party
1	Arizona	126	152
2	Florida	32	85
3	Vermont	230	121
4	Texas	21	215
5	Georgia	200	13

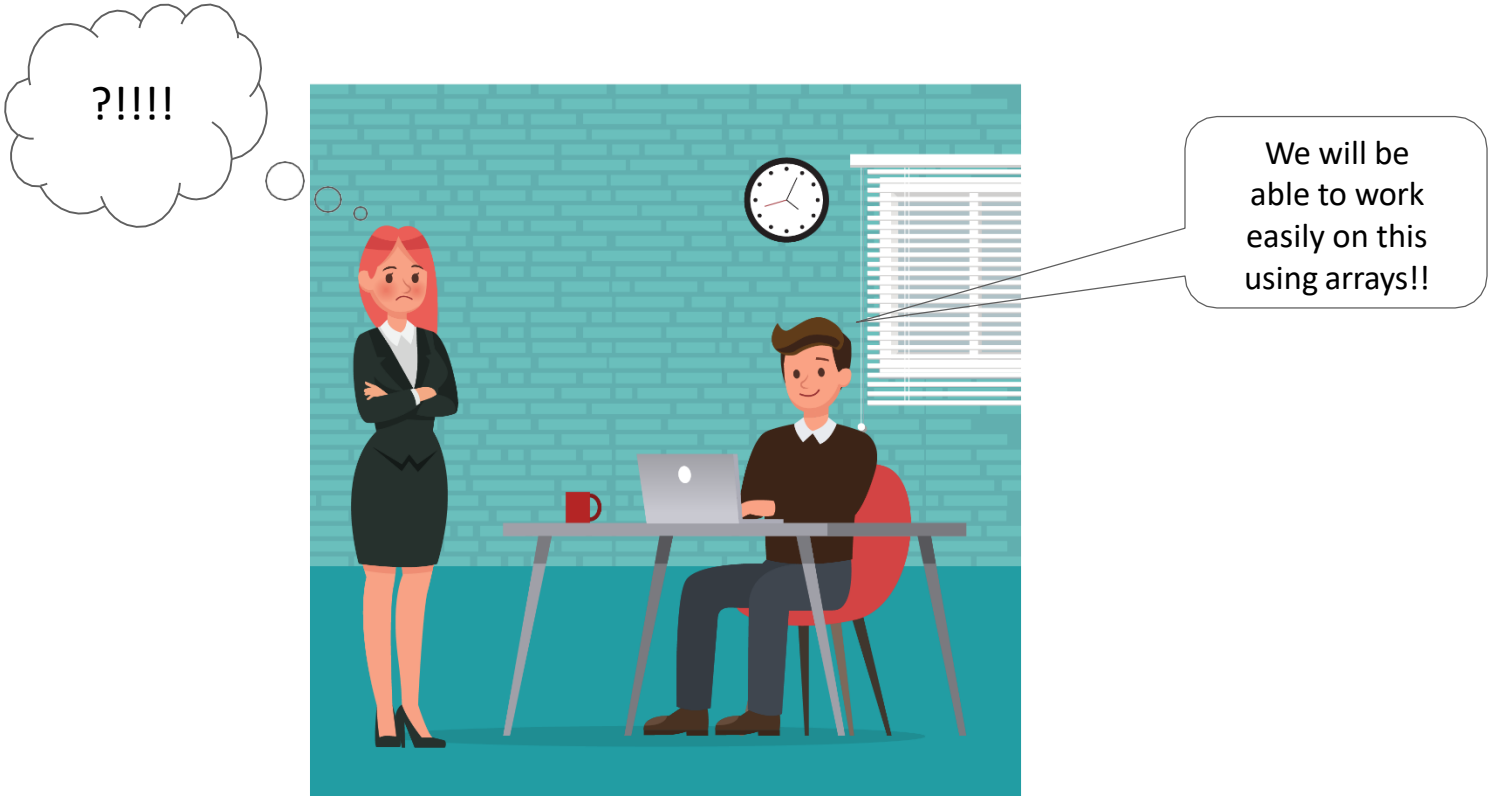
- The data is huge and given for many regions. John needs to find out the wining party
- John takes a small dataset and works on it

Daisy was Confused!

John, How are you planning to go ahead with this?



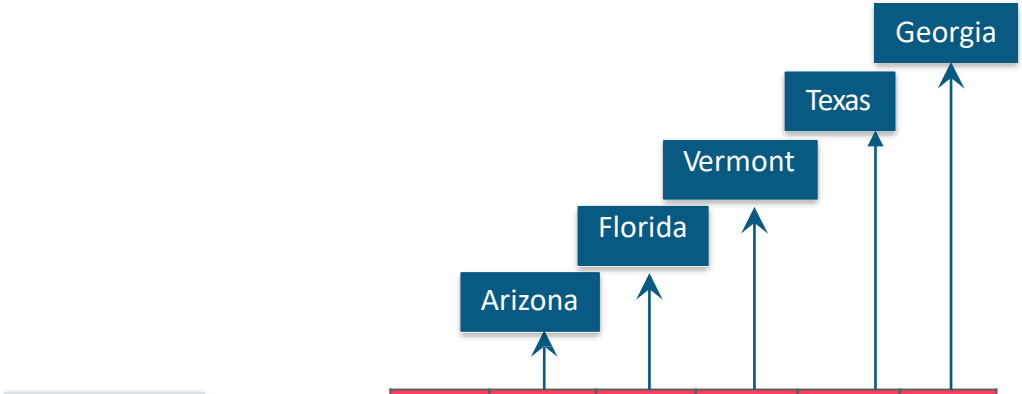
John already had a Plan



John came up with the Solution

And so, John started working on the Project and generated the solution.

John’s Solution



	0	1	2	3	4
0	126	32	230	21	200
1	152	85	121	215	13

$$= \sum_{i=0, j=0}^{j=4} votes(i, j) = 609$$

$$= \sum_{i=1, j=0}^{j=4} votes(i, j) = 586$$

John Presented the Solution

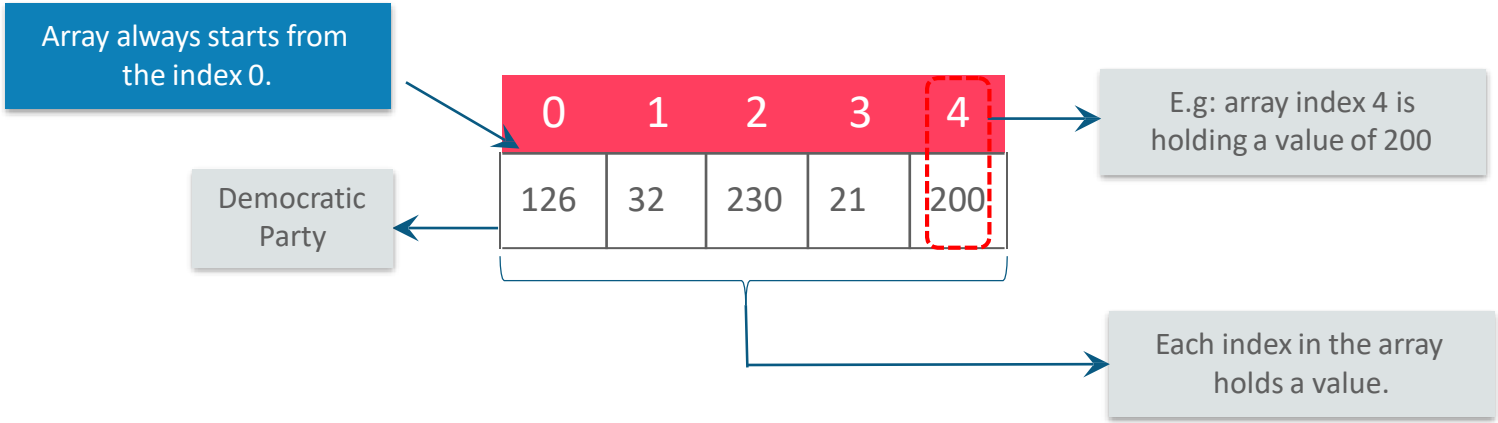
John, I am really very impressed by the way you handled this project. This is a very correct way of dealing with such problems.

John gets a Promotion!

Well done John. You have applied your skills very wisely. I am promoting you.

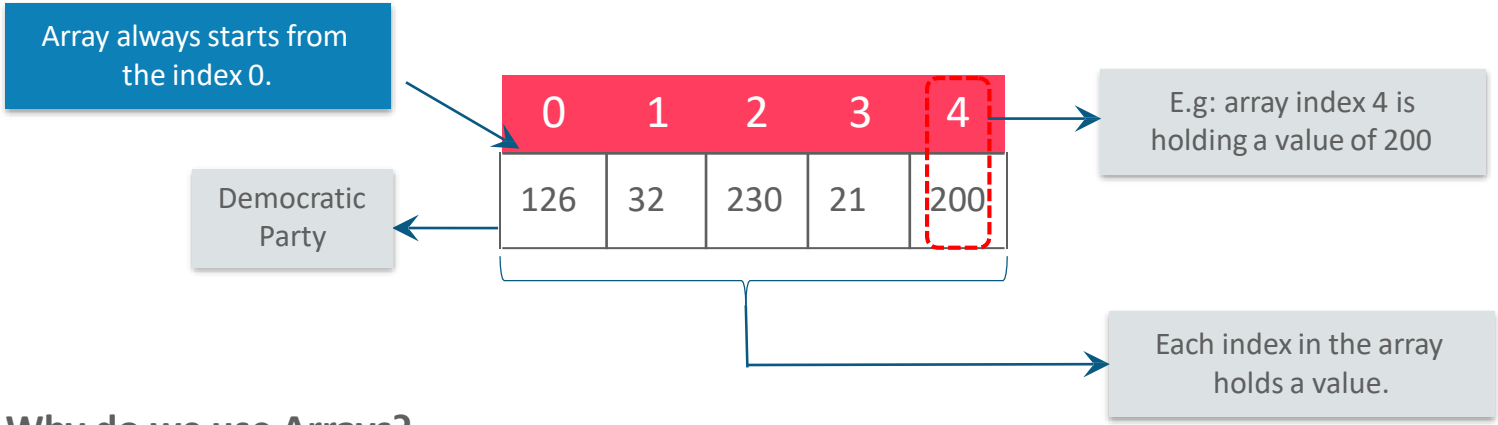
Arrays

- An array is a data structure which holds the sequential elements of the same type
- For Example: Lets create an array for Democratic party from John's project



Arrays

- An array is a data structure which holds the sequential elements of the same type
- For Example: Lets create an array for Democratic party from John's project



Why do we use Arrays?

- Arrays are an important structure to hold data
- Java allows us to hold many objects of the same type using arrays and it can be used with the help of a loop to access the elements by their index

Declaring Arrays

- Array can be declared in multiple ways:

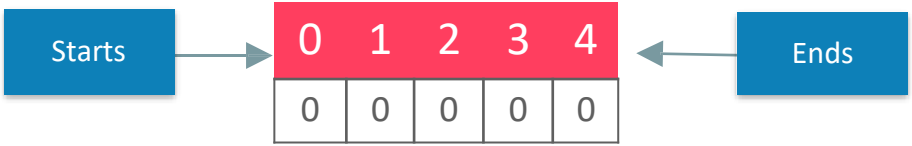
1

data type

size of array

```
int[]a= new int[5];
```

Index has to be given in square brackets



Declaring Arrays

- Array can be declared in multiple ways:

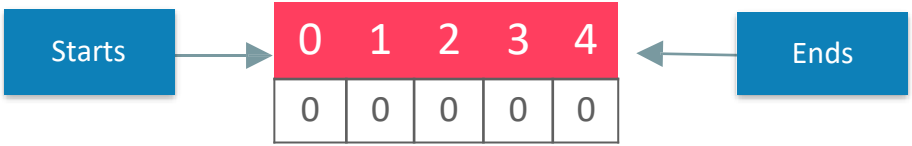
1

data type

size of array

```
int[]a= new int[5];
```

Index has to be given in square brackets



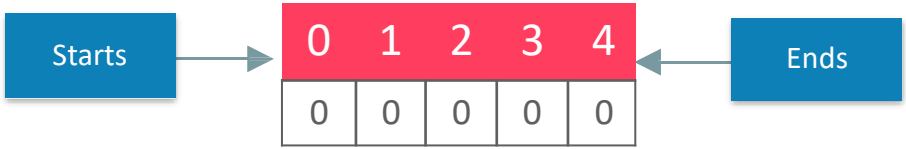
2

data type

size of array

```
int a[]= new int[5];
```

Index has to be given in square brackets



Declaring Arrays

- Array can be declared in multiple ways:

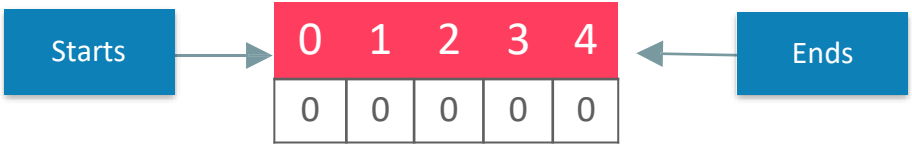
3

data type

size of array

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

Index has to be given in square brackets



Declaring Arrays

- Array can be declared in multiple ways:

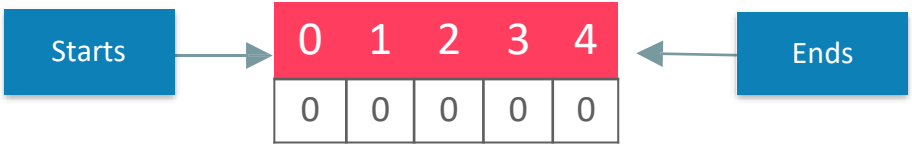
3

data type

size of array

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

Index has to be given in square brackets



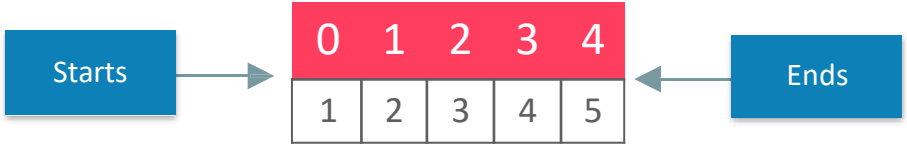
4

data type

size of array

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

Index has to be given in square brackets



Declaring Arrays (contd.)

Arrays can be declared for primitive and non-primitive data types

Declaring Arrays (contd.)

Arrays can be declared for primitive and non-primitive data types

Instead of declaring **int a1, a2, a3, a4....a100**, we can declare **int a[100]**

Declaring Arrays (contd.)

- Arrays can be declared for primitive and non-primitive data types
- Instead of declaring **int a1, a2, a3, a4....a100**, we can declare **int a[100]**
- It is fast and efficient to access an element in an array using the corresponding index without actually traversing through the entire array

Declaring Arrays (contd.)

- Arrays can be declared for primitive and non-primitive data types
- Instead of declaring **int a1, a2, a3, a4....a100**, we can declare **int a[100]**
- It is fast and efficient to access an element in an array using the corresponding index without actually traversing through the entire array
- The size of the array is fixed once it is created

Declaring Arrays (contd.)

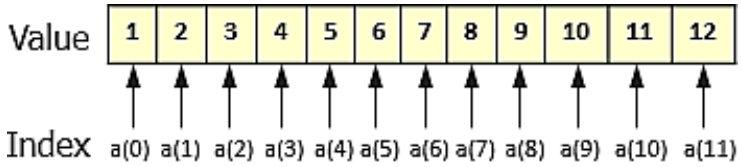
- Arrays can be declared for primitive and non-primitive data types
- Instead of declaring **int a1, a2, a3, a4....a100**, we can declare **int a[100]**
- It is fast and efficient to access an element in an array using the corresponding index without actually traversing through the entire array
- The size of the array is fixed once it is created
- **For Example:** A character array of 10 characters can be declared as

```
char[]ch= new char[10];
```

Types of Arrays

Single Dimensional Array

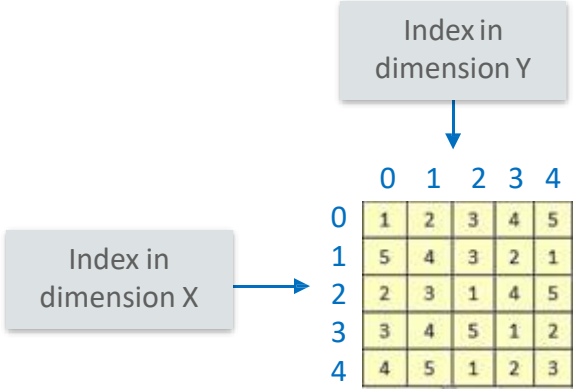
Initialization: `int x[] = new int[12]`



System.out.print(a[5]); Output: 6

Multi Dimensional Array

Initialization: `int x[][] = new int[5][5];`



How Single Dimensional Arrays are used?

- Array can be initialized in this manner also:

```
class arrInit
{
  public static void main (String[] args)
  {
    int i;
    int[] a = {10, 20, 30, 40, 50};
    for (i = 0; i < 5; i++)
    System.out.println (a[i]);
  }
}
```

- This program declares an array 'a' and initializes it with numbers 10, 20, 30, 40 and 50

Multi Dimensional Array

- Multi dimensional arrays can be 2 dimensional as rows and columns similar to a matrix
- It could be 3 dimensional with depth, height and breadth
- Or it could be 'n' dimensional array as per the requirement
- Usually 2 dimensional arrays are used to perform matrix operations
- Like single dimensional arrays, multi dimensional arrays can be of primitive and non-primitive data type

Multi Dimensional Array

- Multi-dimensional arrays can be declared as

```
int [][]a= new int [2][2];
char [][]a= new char[3][2];
float [][]a= new float[5][5];
```

2 x 2 dimensional int array

	0	1
0	1	4
1	4	5

2 x 2 dimensional int array

Multi Dimensional Array

- Multi-dimensional arrays can be declared as

```
int [][]a= new int [2][2];
char [][]a= new char[3][2];
float [][]a= new float[5][5];
```

2 x 2 dimensional int array

3 x 2 dimensional char array

	0	1
0	1	4
1	4	5

2 x 2 dimensional int array

	0	1
0	s	a
1	g	v
2	v	d

3 x 2 dimensional int array

Multi Dimensional Array

- Multi-dimensional arrays can be declared as

```
int [][]a= new int [2][2];
char [][]a= new char[3][2];
float [][]a= new float[5][5];
```

2 x 2 dimensional int array
3 x 2 dimensional char array
5 x 5 dimensional float array

	0	1
0	1	4
1	4	5

2 x 2 dimensional int array

	0	1
0	s	a
1	g	v
2	v	d

3 x 2 dimensional int array

	0	1	2	3	4
0	2.2	3.4	5.0	3.3	1.2
1	7.8	9.0	1.1	2.9	5.5
2	2.0	3.0	7.8	9.8	9.9
3	5.7	6.6	8.8	5.3	2.7
4	1.8	4.4	7.6	1.0	1.1

5 x 5 dimensional float array

In Class Questions

1. How many bytes are allocated for the array x? int x[][] = new int[5][5];
2. Can we do multi dimensional array operations in single dimensional array? If yes, then why do we require multi dimensional array?

In Class Questions - Solutions

1. How many bytes are allocated for the array x? int x[][] = new int[5][5];

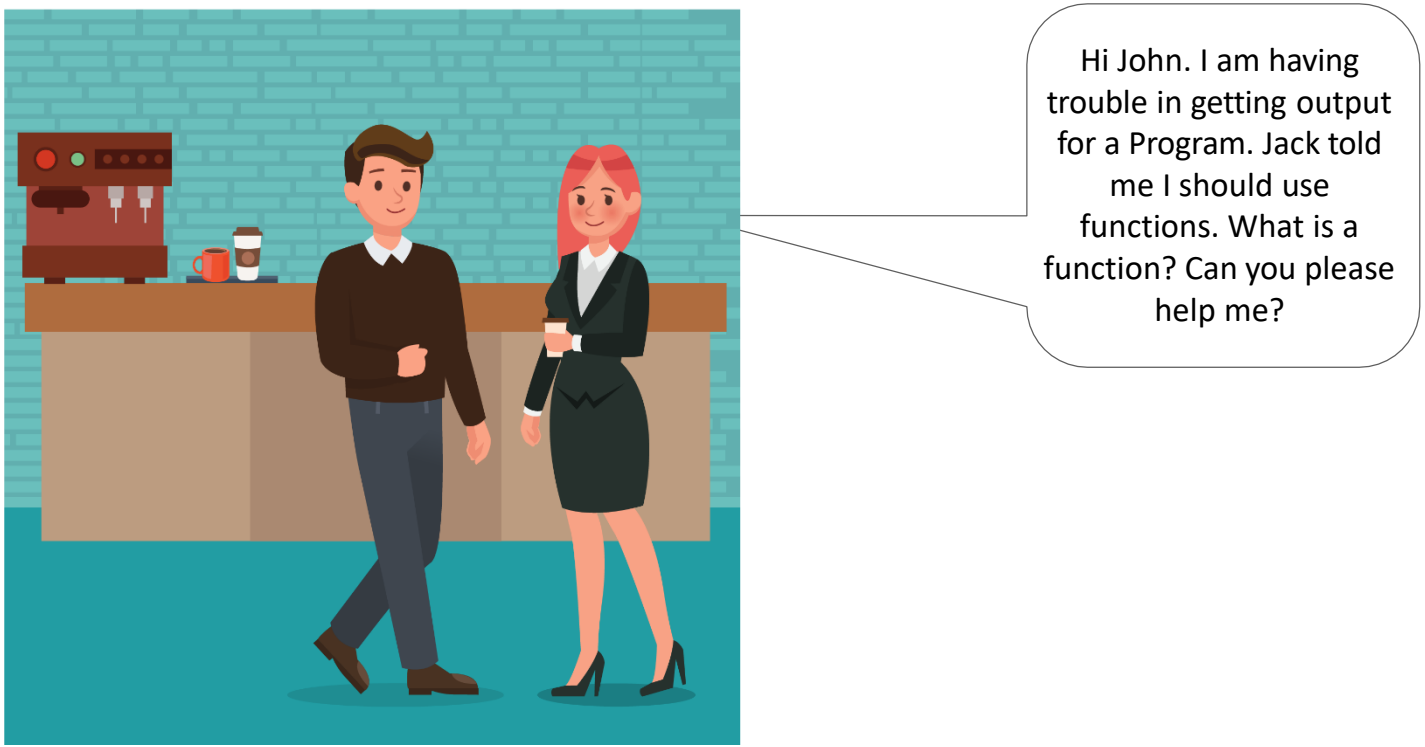
Solution: An int element takes 4 bytes and array x can store 25 int elements. Hence total number of bytes taken by x is 25*4 = 100

2.Can we do multi dimensional array operations in single dimensional array? If yes, then why do we require multi dimensional array?

Solution: We can perform all the operations of multi dimensional array in single dimensional array but when the data has more than one dimension, it will be easier to perform operations in multi dimensional array.

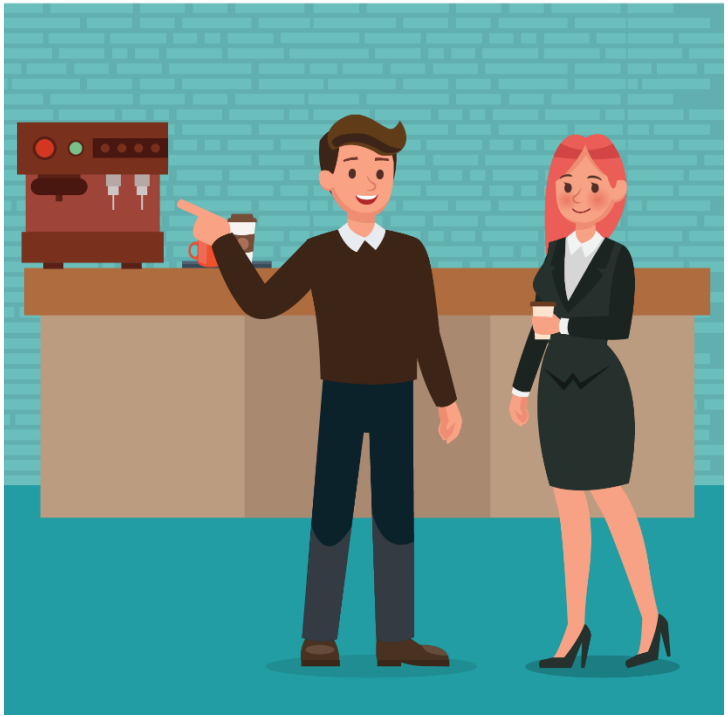
Function

Daisy seeks Help!

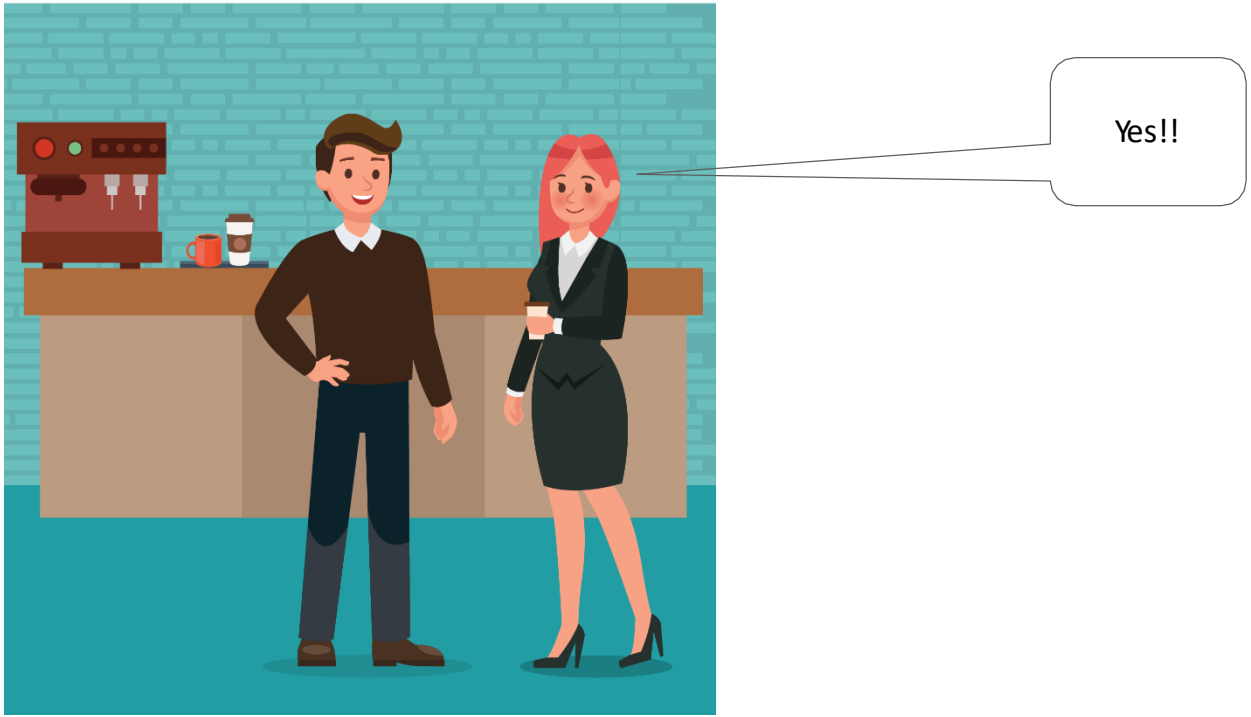


John helps Daisy

Yes sure. Do you see the coffee machine on the table?



John helps Daisy



John explain Functions

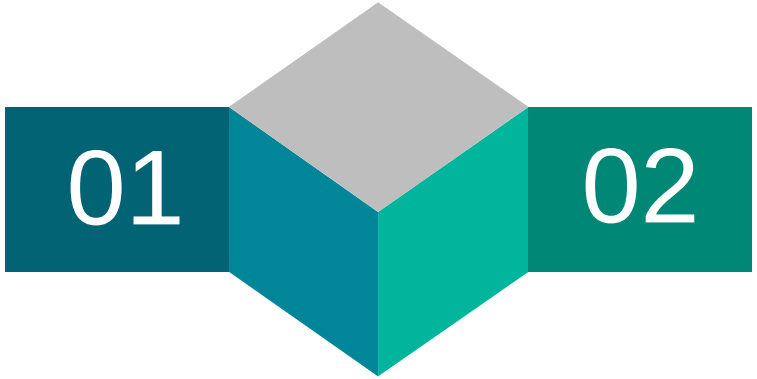
You just press a button and get coffee within seconds. There are a lot of processes running at the backend, but you do not implement them each time you need coffee. Instead you only get the output.

John explain Functions

Functions work in a similar way. It helps you save your time and effort while writing programs in Java. Let's learn in details.

Functions

A function/method has group of statements to be executed



A function has a task to perform

A function is declared as -

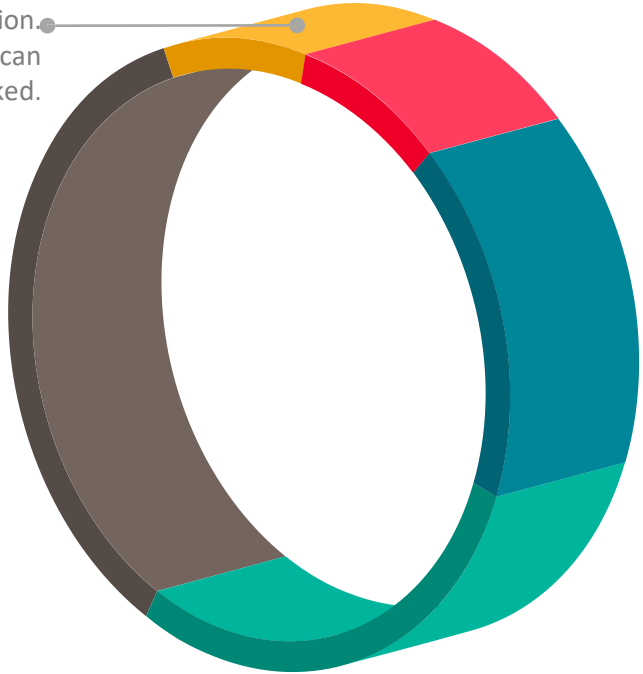
```
Scope specifier modifier  return type  function name (arguments)
```

```
public static void say Hello(int a, int b)
```

Why do we use Functions ?

Function name

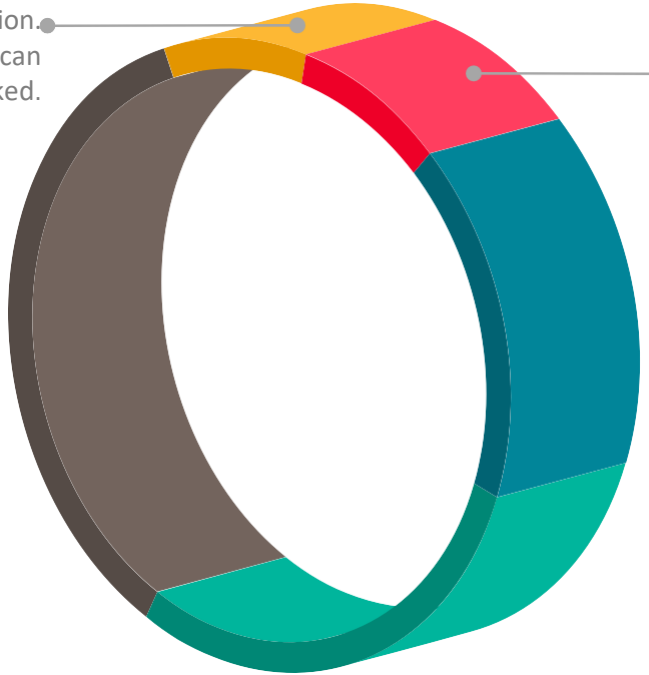
It is the name given to the function. Using function name, functions can be invoked.



Why do we use Functions ?

Function name

It is the name given to the function. Using function name, functions can be invoked.



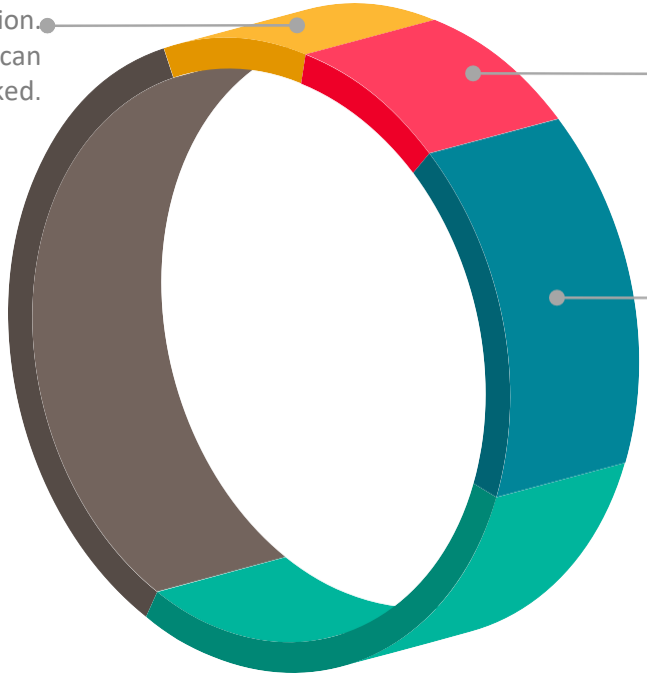
Scope specifier

This option specifies the visibility of the function. It could be public, private, protected or default (package)

Why do we use Functions ?

Function name

It is the name given to the function. Using function name, functions can be invoked.



Scope specifier

This option specifies the visibility of the function. It could be public, private, protected or default (package)

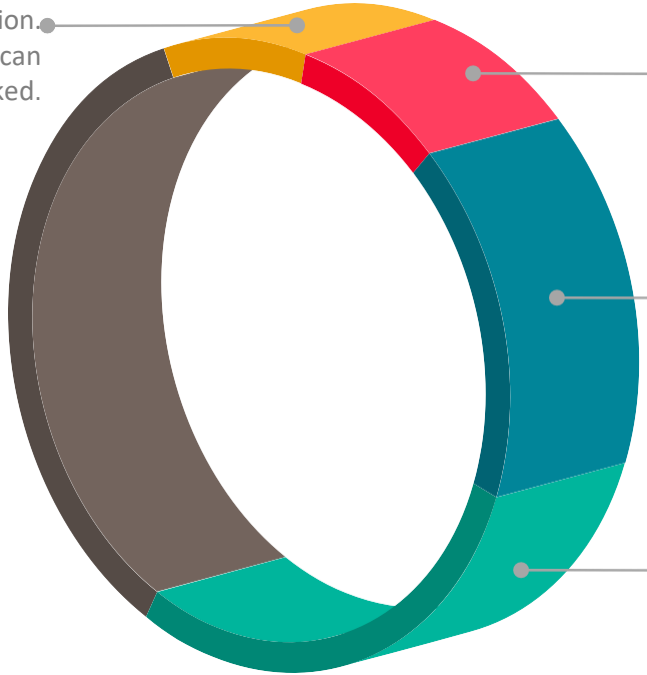
Modifier

This will change the meaning of the function. Some of the options could be final, static, native, synchronized etc.

Why do we use Functions ?

Function name

It is the name given to the function. Using function name, functions can be invoked.



Scope specifier

This option specifies the visibility of the function. It could be public, private, protected or default (package)

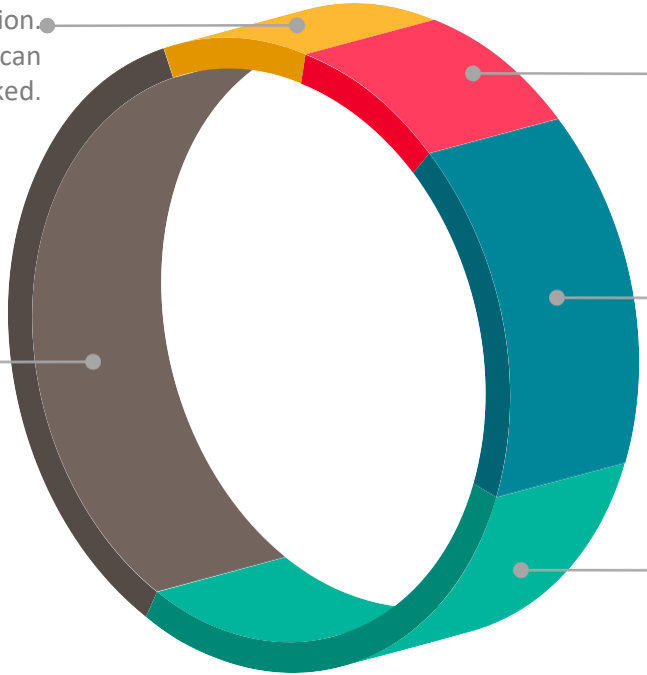
Modifier

This will change the meaning of the function. Some of the options could be final, static, native, synchronized etc.

Return type

This option specifies the data that will be returned by the function. It could be primitive data type or object.

Why do we use Functions ?



Function name

It is the name given to the function. Using function name, functions can be invoked.

Scope specifier

This option specifies the visibility of the function. It could be public, private, protected or default (package)

Modifier

This will change the meaning of the function. Some of the options could be final, static, native, synchronized etc.

Arguments

This is optional but can be used. Using this, one can pass values to the function for computation.

Return type

This option specifies the data that will be returned by the function. It could be primitive data type or object.

Different kinds of Functions

- We use functions to provide different kinds of functionality

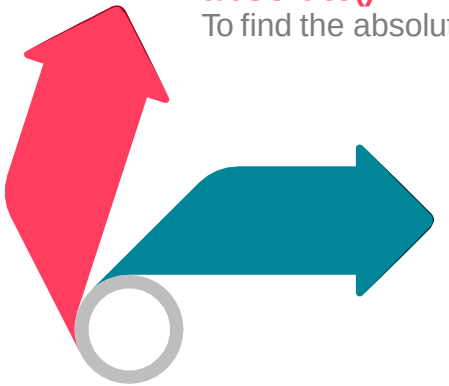
For example:

absolute()
To find the absolute value of a number

Different kinds of Functions

- We use functions to provide different kinds of functionality

For example:



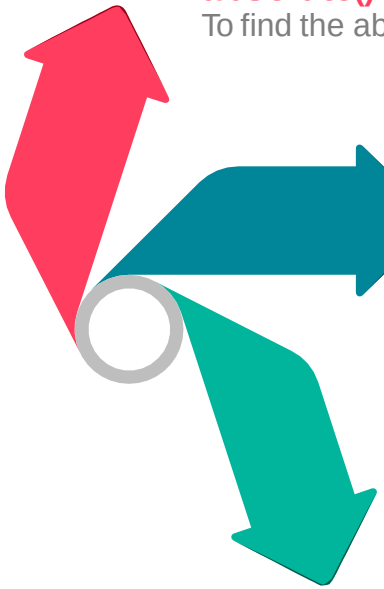
absolute()
To find the absolute value of a number

square()
To find the square of a number

Different kinds of Functions

- We use functions to provide different kinds of functionality

For example:

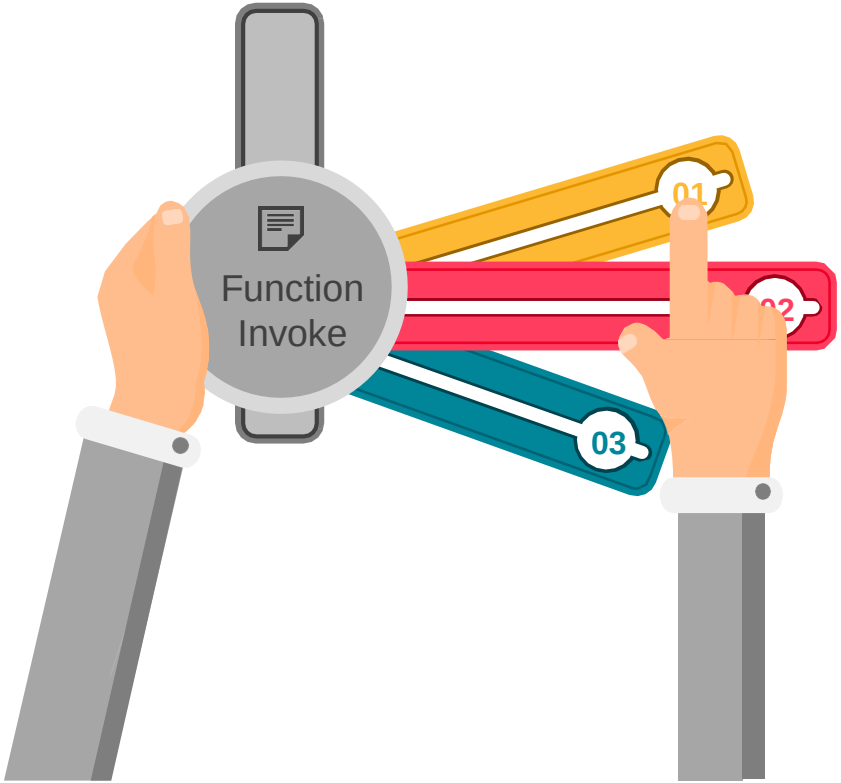


absolute()
To find the absolute value of a number

square()
To find the square of a number

sqrt()
To find the square root of a number

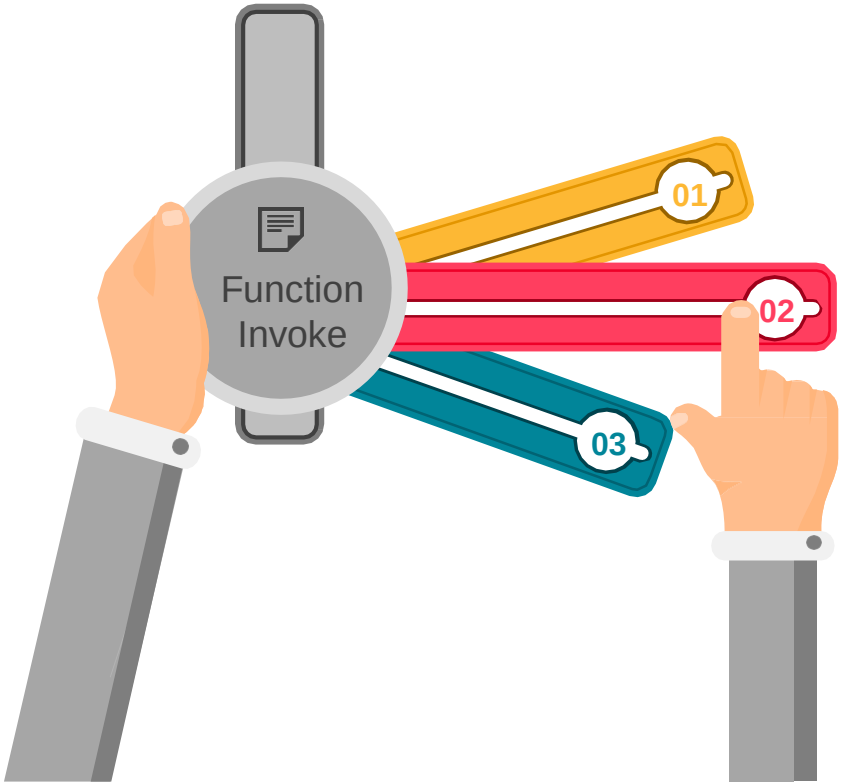
What happens when a Function is Invoked?



When a function is called from the main method, main method() address is stored on top of the stack

01

What happens when a Function is Invoked?



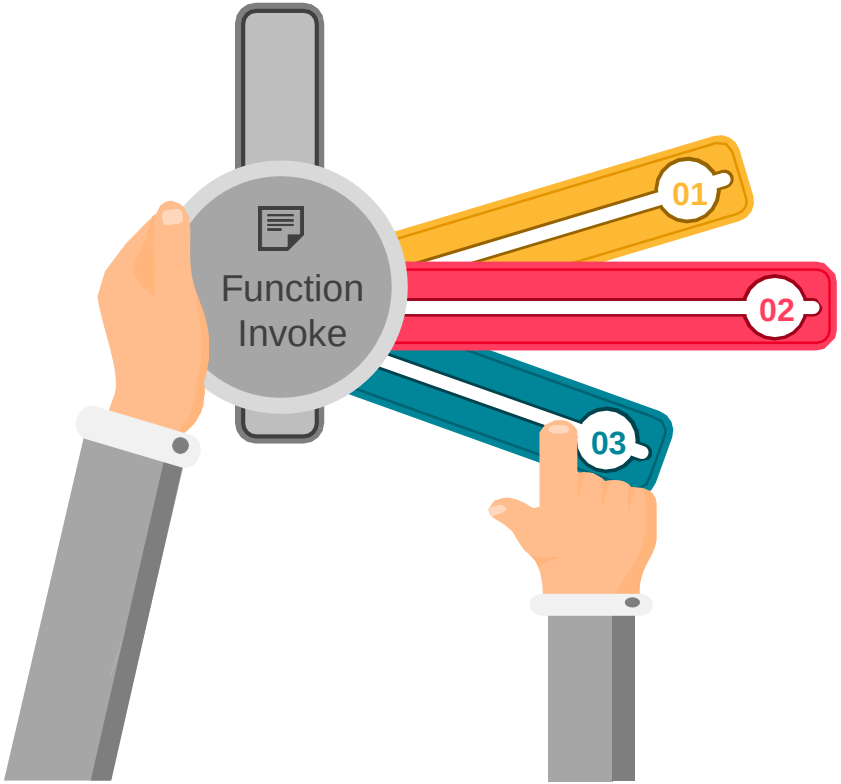
When a function is called from the main method, main method() address is stored on top of the stack

01

Control jumps to execute the function

02

What happens when a Function is Invoked?



When a function is called from the main method, main method() address is stored on top of the stack

01

Control jumps to execute the function

02

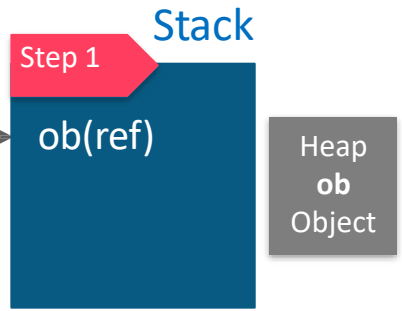
After executing the function, main method address is popped from the stack and main method execution resumes

03

What happens when a Function is Invoked?

*class1.java

```
1
2  public class class1 {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          class2 ob = new class2();
8
9          ob.fun();
10     }
11
12 }
```



*class2.java

```
1
2  public class class2 {
3
4      public void fun() {
5          // TODO Auto-generated method stub
6          int x= 10;
7      }
8
9  }
10
11
12
```

Ob(ref)

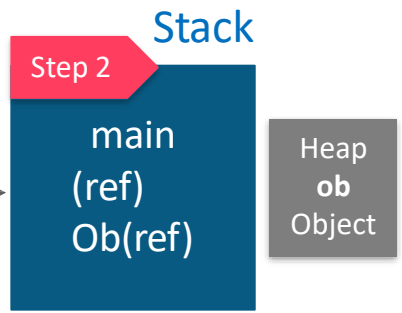
What happens when a Function is Invoked? (contd.)

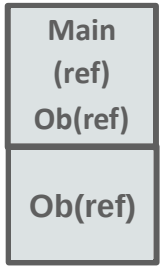
*class1.java

```
1
2  public class class1 {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          class2 ob = new class2();
8
9          ob.fun();
10     }
11
12 }
```

*class2.java

```
1
2  public class class2 {
3
4      public void fun() {
5          // TODO Auto-generated method stub
6          int x= 10;
7      }
8
9  }
10
11
12
```





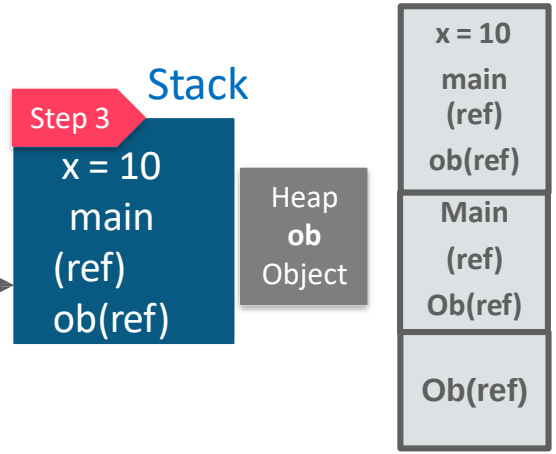
What happens when a Function is Invoked? (contd.)

*class1.java

```
1
2  public class class1 {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          class2 ob = new class2();
8
9          ob.fun();
10     }
11
12 }
```

*class2.java

```
1
2  public class class2 {
3
4      public void fun() {
5          // TODO Auto-generated method stub
6          int x= 10;
7      }
8
9  }
10
11
12
```



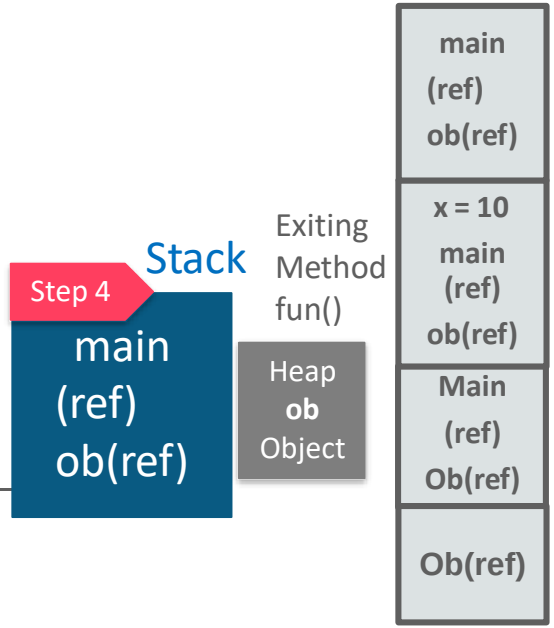
What happens when a Function is Invoked? (contd.)

*class1.java

```
1
2  public class class1 {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          class2 ob = new class2();
8
9          ob.fun();
10     }
11
12 }
```

*class2.java

```
1
2  public class class2 {
3
4      public void fun() {
5          // TODO Auto-generated method stub
6          int x= 10;
7      }
8
9  }
10
11
12
```



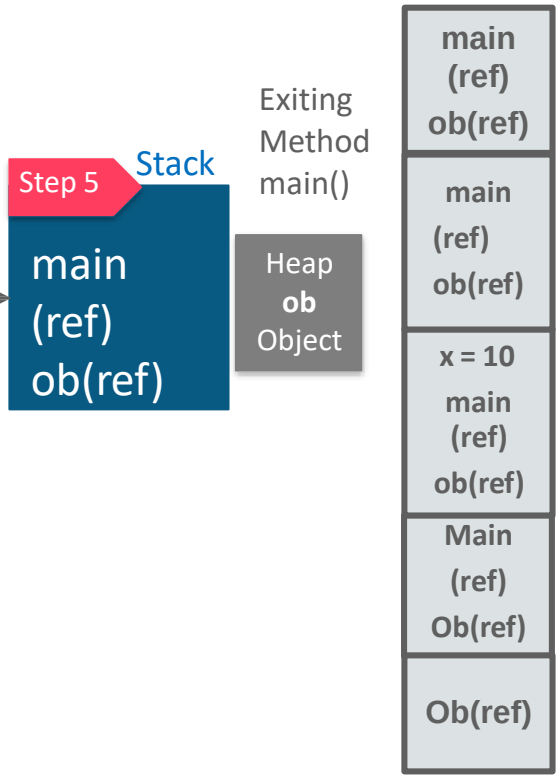
What happens when a Function is Invoked? (contd.)

*class1.java

```
1
2  public class class1 {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          class2 ob = new class2();
8
9          ob.fun();
10     }
11
12 }
```

*class2.java

```
1
2  public class class2 {
3
4      public void fun() {
5          // TODO Auto-generated method stub
6          int x= 10;
7      }
8
9  }
10
11
12
```



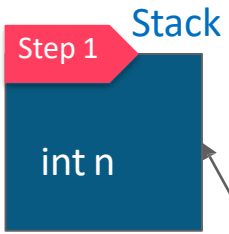
Sample Program on Functions

- Program to find square of a given number

```
public class testFunctions
{
   //function to return square of a given number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a number...");
     int n = sc.nextInt();
     int result = testFunctions.square (n);
     System.out.println ("Square of " + n + " is " + result);
   }
}
```

Sample Program on Functions(contd.)

- Program to find square of a given number




```
public class testFunctions
{
   //function to return square of a given number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a number...");
     int n = sc.nextInt();
     int result = testFunctions.square (n);
     System.out.println ("Square of " + n + " is " + result);
   }
}
```

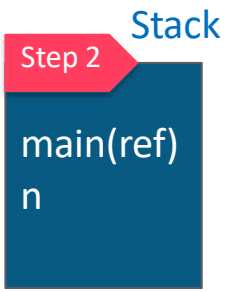
Step 2

Sample Program on Functions

- Program to find square of a given number

```
public class testFunctions
{
   //function to return square of a given number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a number...");
     int n = sc.nextInt();
     int result = testFunctions.square (n);
     System.out.println ("Square of " + n + " is " + result);
   }
}
```

Step 2



Sample Program on Functions (Contd.)

- Program to find square of a given number

```
public class testFunctions
{
   //function to return square of a given number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a number...");
     int n = sc.nextInt();
     int result = testFunctions.square (n);
     System.out.println ("Square of " + n + " is " + result);
   }
}
```

Step 2

Step 3

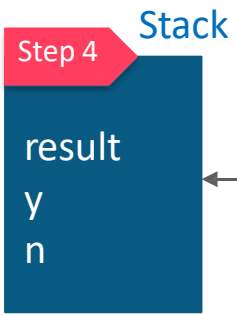
Stack

y
x
main(ref)
n

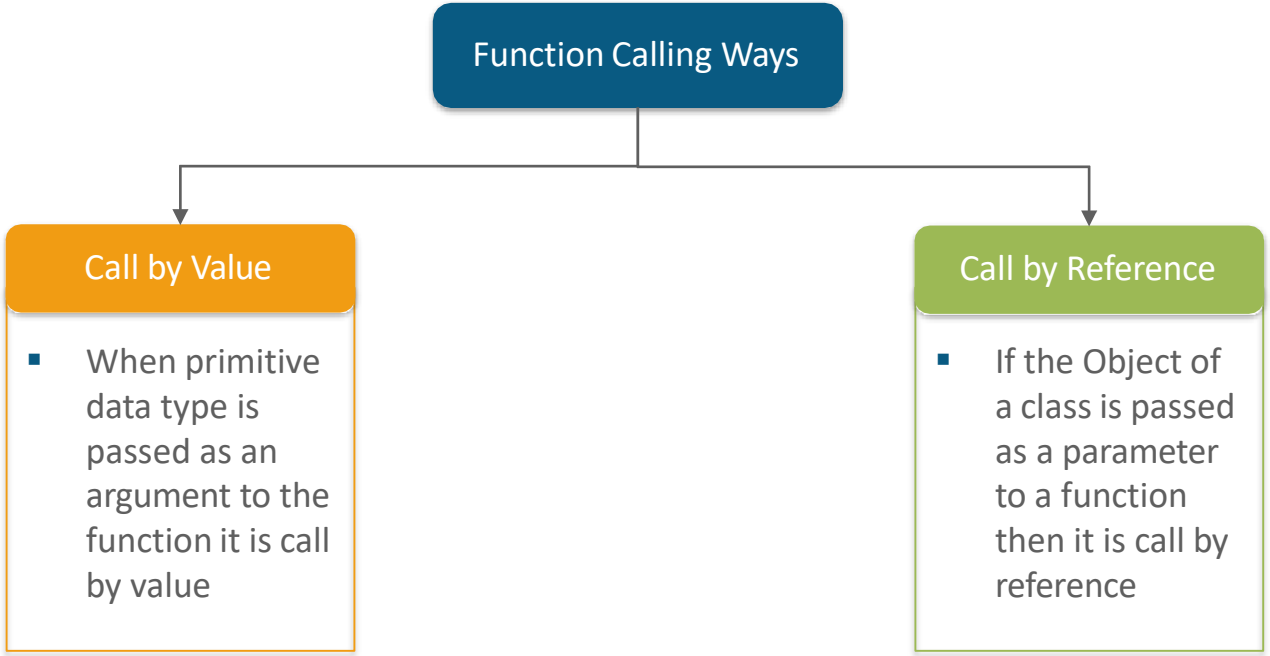
Sample Program on Functions (Contd.)

- Program to find square of a given number

```
public class testFunctions
{
   //function to return square of a given number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a number...");
     int n = sc.nextInt();
     int result = testFunctions.square (n);
     System.out.println ("Square of " + n + " is " + result);
   }
}
```



Function Calling Ways



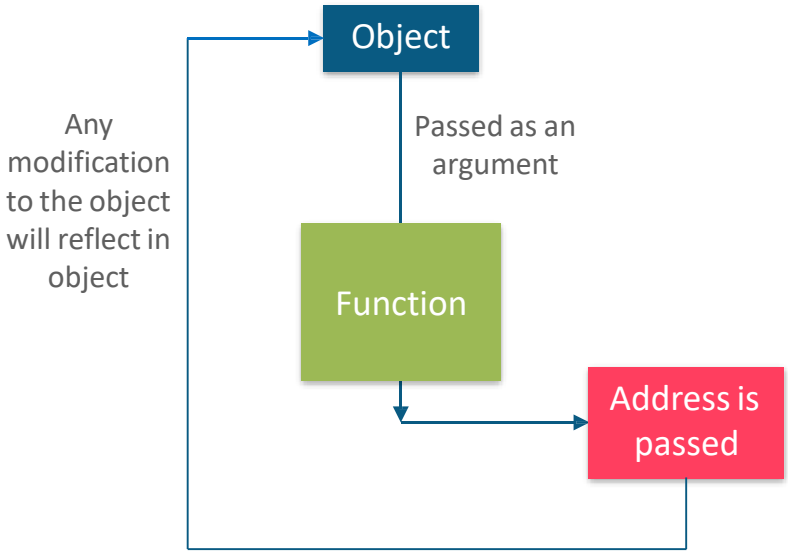
Call by Value

```
public class pow
{
   //function to return square of a given
number.
      int static square (int x)
   {
      int y = x * x;
      return y;
   }

public static void main (String[] args)
   {
     Scanner sc = new Scanner (System.in);
     System.out.println ("Please enter a
number...");
     int n = sc.ne xtInt();

     int result = pow.square (n);

     System.out.println ("Square of " + n + "
is " + result);
   }
}
```

- When value of the primitive data type is passed as an argument from the calling function, data is being sent to the function.
- At the receiving end, a new variable is created and the data is copied. This is call by value.
- All the programs we have done till now is done using call by value.

Call by Reference

- When an object is passed as an argument to the function, its address is being passed so that any modification to the object will reflect in the object itself. This is call by reference.



Call by Reference

```
public class callByReference
{
    int data=20;
    public void functionDemo (callByReference
test)
      {
        test.data = test.data * 2;
      }
     public static void main (String[] args){
        callByReference c1 = new
callByReference();
        System.out.println ("Before calling
the function : "+test.data);

        c1.functionDemo (c1);

        System.out.println ("After calling
the function : "+test.data);
       }
}
```

Out of the program is

Call by Reference

```
public class callByReference
{
    int data=20;
    public void functionDemo (callByReference
test)
      {
        test.data = test.data * 2;
      }
     public static void main (String[] args){
        callByReference c1 = new
callByReference();
        System.out.println ("Before calling
the function : "+c1.data);

        c1.functionDemo (c1);

        System.out.println ("After calling
the function : "+c1.data);
       }
}
```

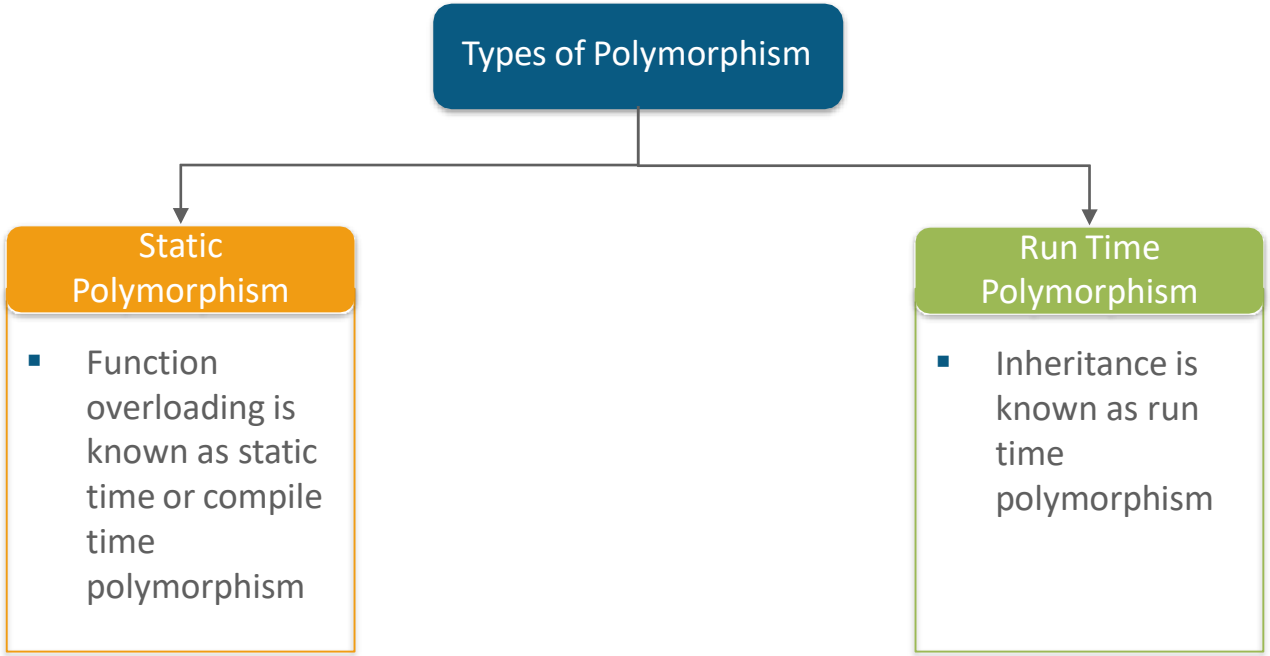
Out of the program is

Before calling the function: 20

After calling the function: 40

Polymorphism

- Polymorphism means the system behaves differently in different programming context



Function Overloading

- Functions by different operators in a Calculator

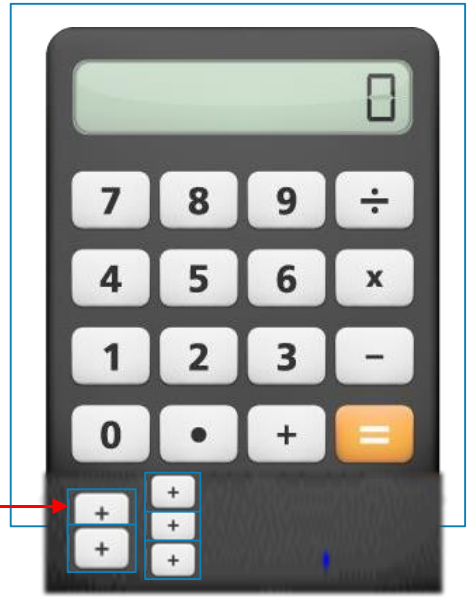

```
÷ can divide:
    → two integers
    → two decimals
```

```
+ can add:
    → two integers
    → three integers
    → two decimals
    → three decimals
```

Function Overloading (contd.)

- Addition is done using a function. If we did not have the concept of method overloading then the calculator will have multiple plus buttons for the below tasks (each for a task):

```
→ two integers     → +
→ three integers   → ++
→ two decimals     → +++
→ three decimals   → ++++
```



Function Overloading (Contd.)

- If many functions / methods have the same name but different arguments it is called Function Overloading

For Example:

If the user calls this method with integer data then the square method with integer argument will be invoked

```
int square (int x);
```

Method name is same

```
double square (double x);
```

If the user calls this method with double data then the square method with double argument will be invoked.

Function Overloading (Contd.)

- Function overloading does not depends on return type. It is differentiated only by the arguments of the function.

For Example:

If the user creates two methods with same name and argument list but makes change in only return type then the method will not be overloaded and will result in syntax error

```
int square (int x);
```

```
double square (int x);
```

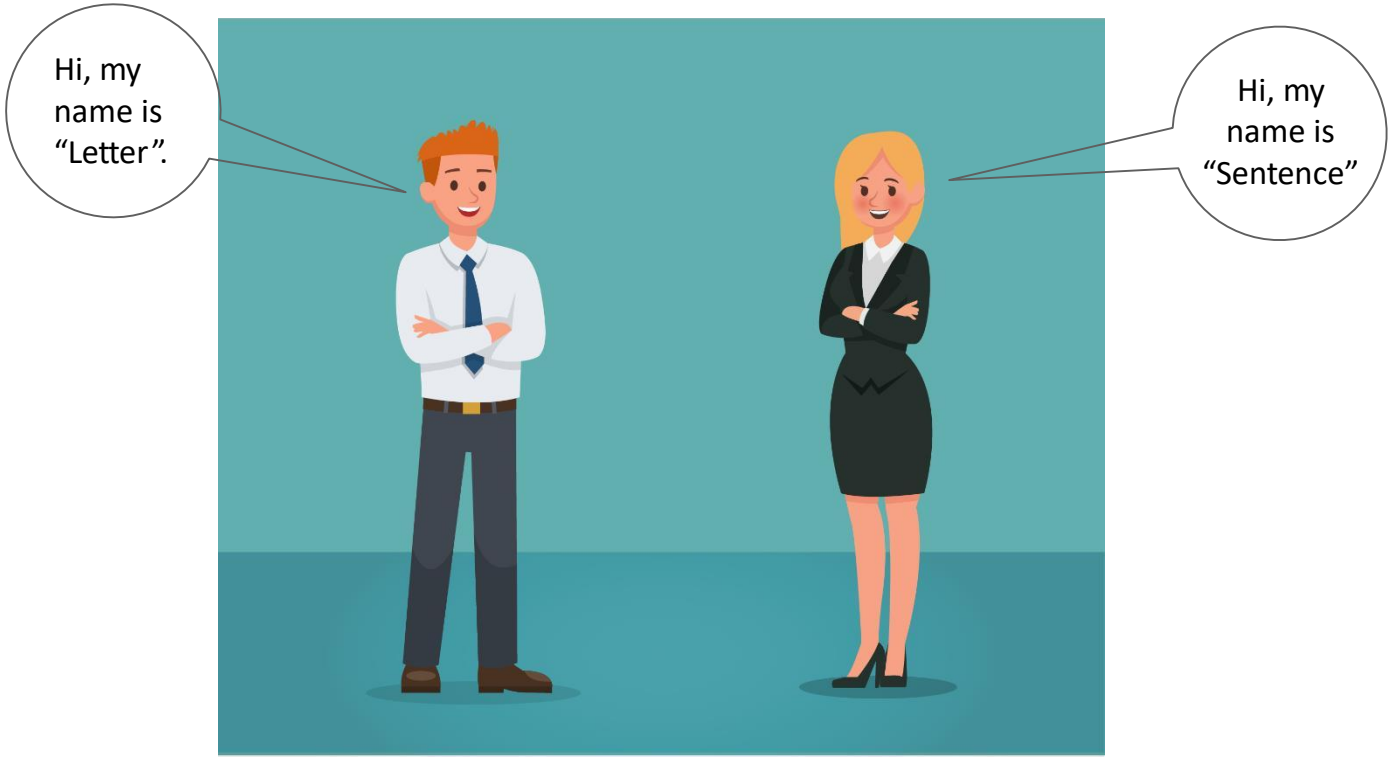
Return type is different

Method name and argument is same

Program on Function Overloading

```
public class FunctionOverloading
{
  static int square (int x)
   {
    return x * x;
   }
  static double square (double x)
   {
    return x * x;
   }
public static void main (String[] args)
   {
    int y = FunctionOverloading.square(10);
    double z = FunctionOverloading.square (12.12);
    System.out.println("Square of 10 is " + y);
    System.out.println("Square of 12.12 is " + z);
   }
}
```

Meet Mr. Letter and Miss Sentence!



Mr. Letter is from “char datatype”

Miss Sentence is sad!

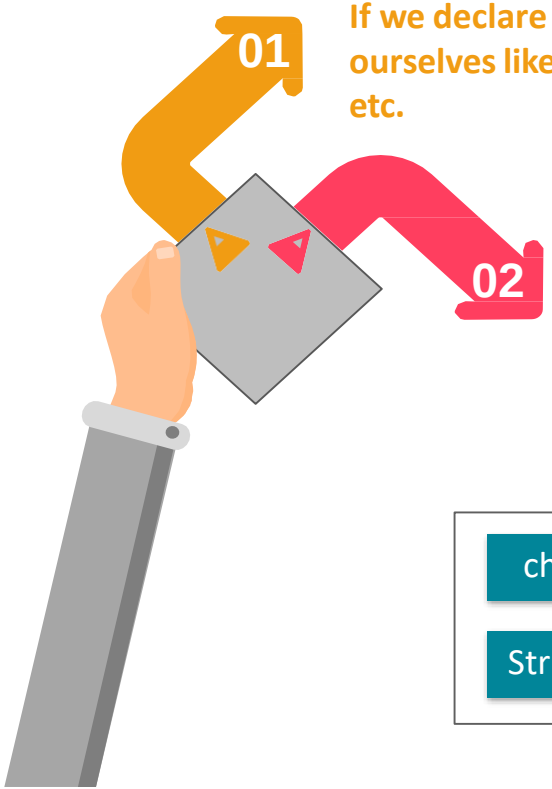
Miss Sentence has her own Place in Java

Come on, don't be sad. Everyone has their own place in Java. For you it is the "String datatype"

Why do we use String?

If we declare character arrays then we have to write string handling methods by ourselves like length of the string, copying string, getting part of the string (sub string) etc.

Why do we use String?

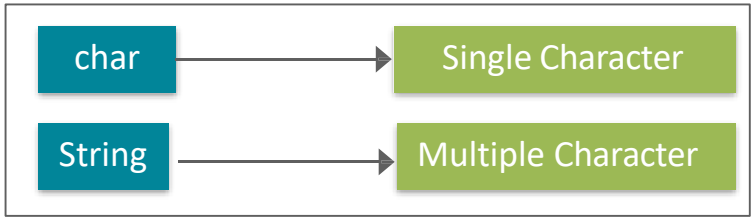


01

If we declare character arrays then we have to write string handling methods by ourselves like length of the string, copying string, getting part of the string (sub string) etc.

02

String class is given in Java to handle multiple characters / Strings and many methods are given in this class to handle the Strings



String

- String is a class in Java to store string data
- You can assign string data simply by defining the string object and assign the string like this:

```
String str;
String str = “Dursikshya”;
String str = new String();
String str = new String(“Dursikshya");
```

- You can concatenate two string using + symbol.

For Example:

```
String str1 = “Hello”;
String str2 = “World”;
String str3 = str1 + str2;
System.out.println (“Concatenated
string is : “+str3);
```

Out of the program is

String

- String is a class in Java to store string data
- You can assign string data simply by defining the string object and assign the string like this:

```
String str;
String str = “Dursikshya”;
String str = new String();
String str = new String(“Dursikshya");
```

- You can concatenate two string using + symbol.

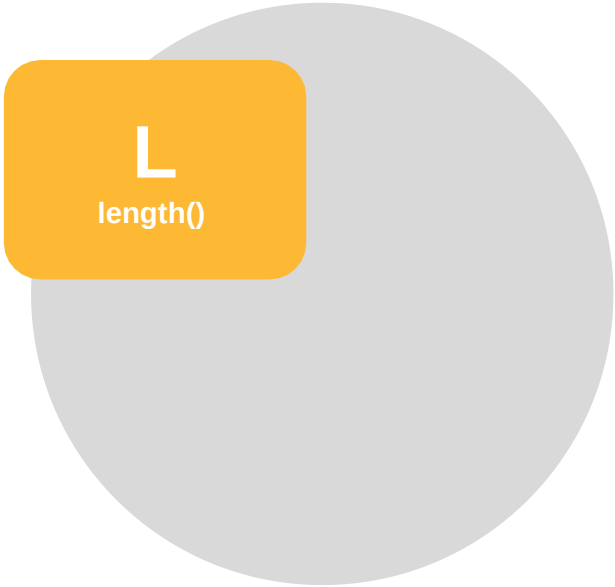
For Example:

```
String str1 = “Hello”;
String str2 = “World”;
String str3 = str1 + str2;
System.out.println (“Concatenated
string is : “+str3);
```

Out of the program is

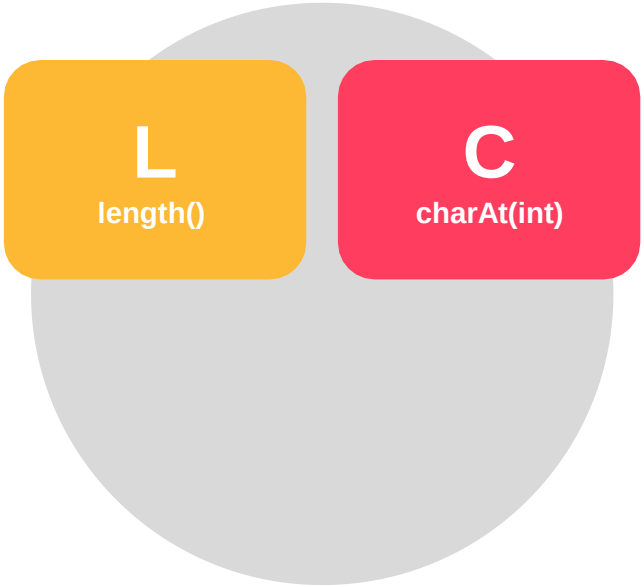
Appended string is: HelloWorld

String Functions



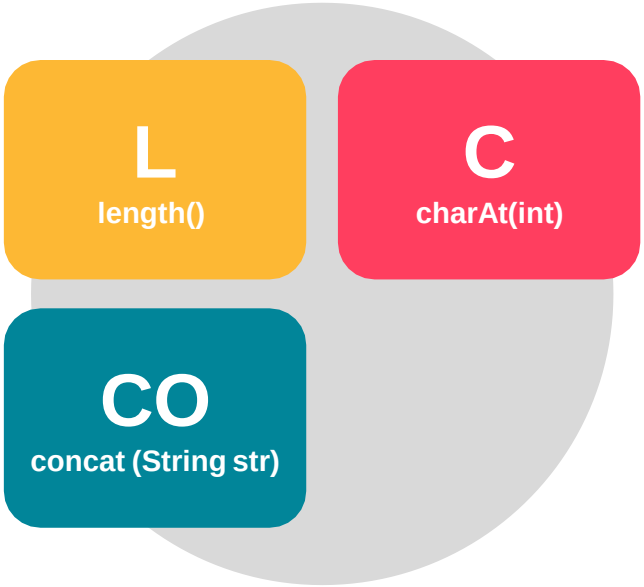
length()
Returns the length of the string

String Functions (Contd.)



length()	charAt(int)
Returns the length of the string	Returns a character at the specified position. Index of the String starts from 0

String Functions (Contd.)



length()

Returns the length of the string

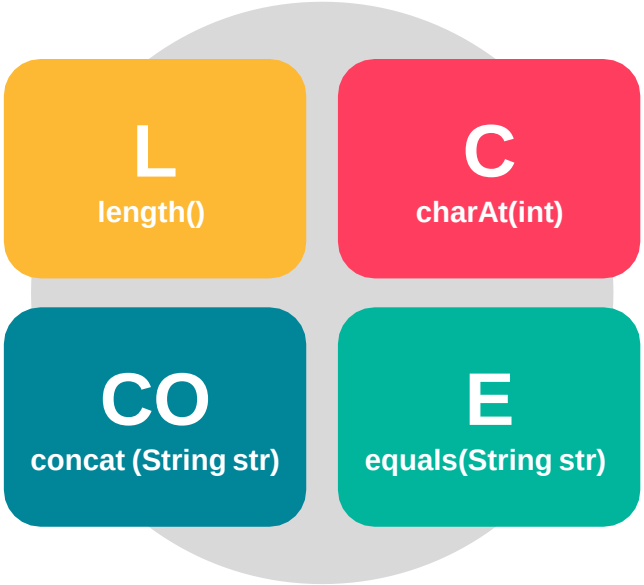
charAt(int)

Returns a character at the specified position. Index of the String starts from 0

concat (String str)

It concatenates the with the string object. It is same as using '+' operator for concatenation

String Functions (Contd.)



length()

Returns the length of the string

charAt(int)

Returns a character at the specified position. Index of the String starts from 0

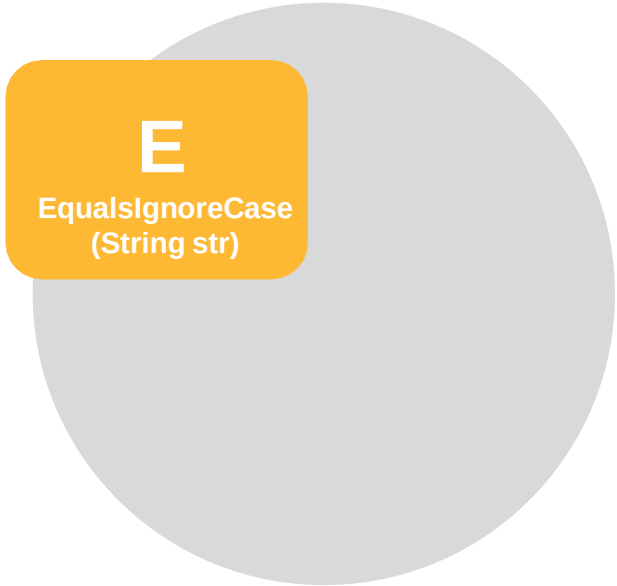
concat (String str)

It concatenates the with the string object. It is same as using '+' operator for concatenation

equals(String str)

Checks whether string object and str and return true if they are same else returns false

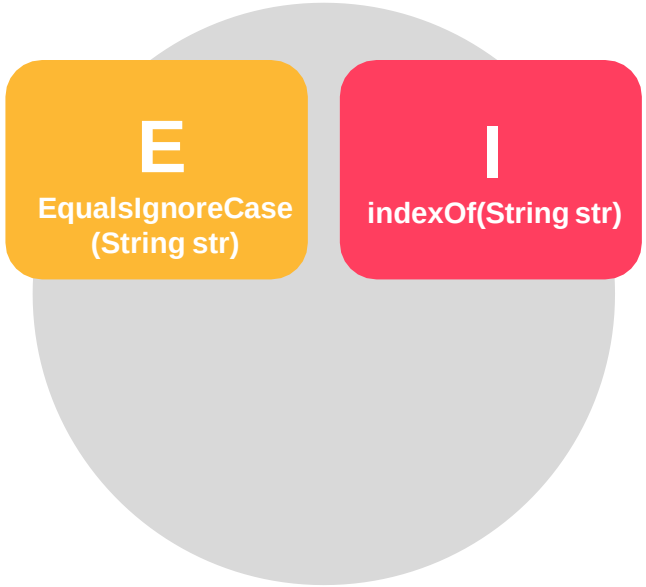
String Functions (Contd.)



EqualsIgnoreCase (String str)

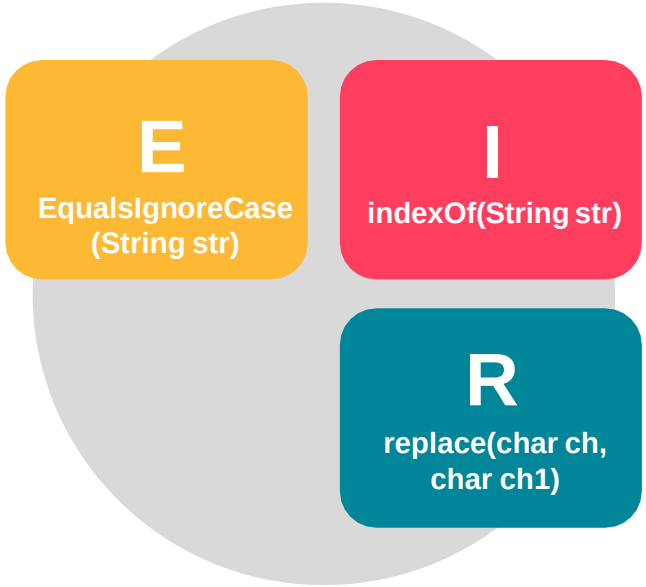
Same as other function except that this function ignores the case and checks for the equality of the strings

String Functions (Contd.)



EqualsIgnoreCase (String str)	indexOf(String str)
Same as other function except that this function ignores the case and checks for the equality of the strings	Returns the index of the specified string in the string object

String Functions (Contd.)



EqualsIgnoreCase (String str)

Same as other function except that this function ignores the case and checks for the equality of the strings

indexOf(String str)

Returns the index of the specified string in the string object

replace(char ch, char ch1)

Replaces the character ch with

Character ch1 in the string

String Functions (Contd.)

E
EqualsIgnoreCase (String str)

I
indexOf(String str)

L
lastIndexOf(String str)

R
replace(char ch, char ch1)

EqualsIgnoreCase (String str)

Same as other function except that this function ignores the case and checks for the equality of the strings

indexOf(String str)

Returns the index of the specified string in the string object

replace(char ch, char ch1)

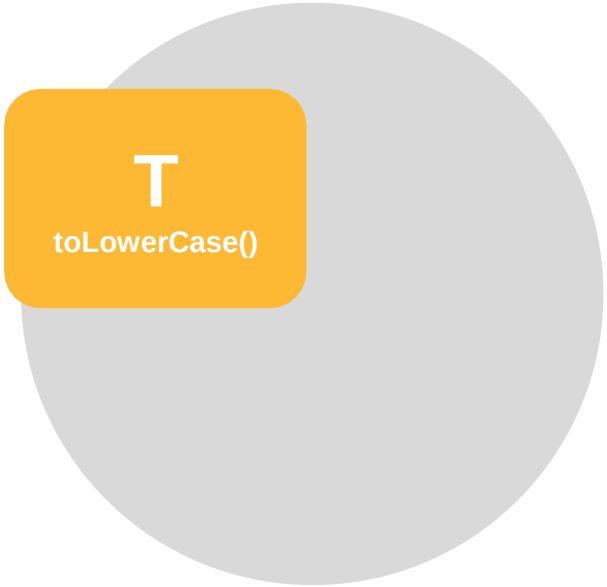
Replaces the character ch with

Character ch1 in the string

lastIndexOf(String str)

Checks whether string object and str and return true if they are same else returns false

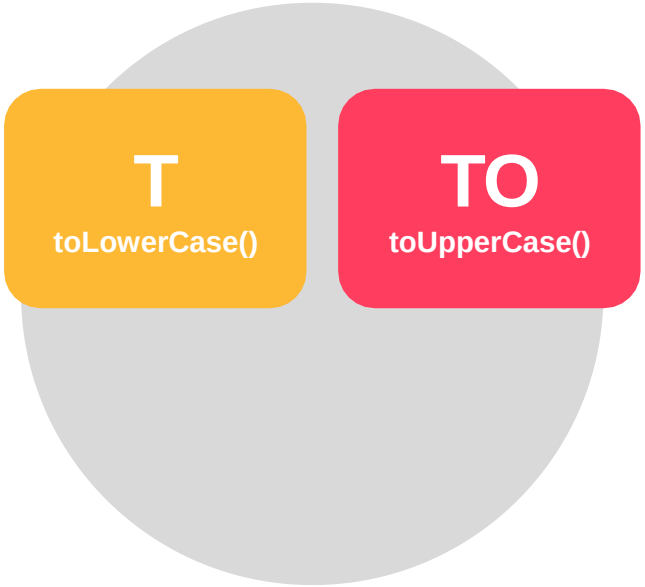
String Functions (Contd.)



toLowerCase()

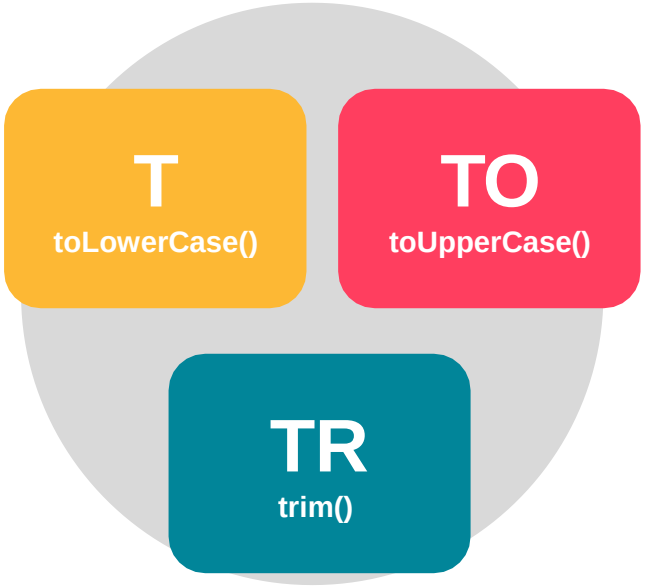
Converts the given string to lowercase

String Functions (Contd.)



toLowerCase()	toUpperCase()
Converts the given string to lowercase	Converts the given string to uppercase

String Functions (Contd.)



toLowerCase()

Converts the given string to lowercase

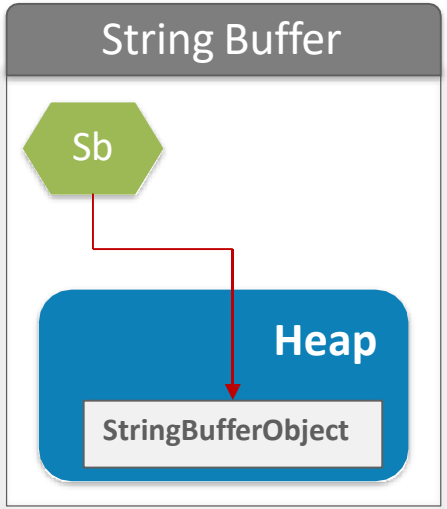
toUpperCase()

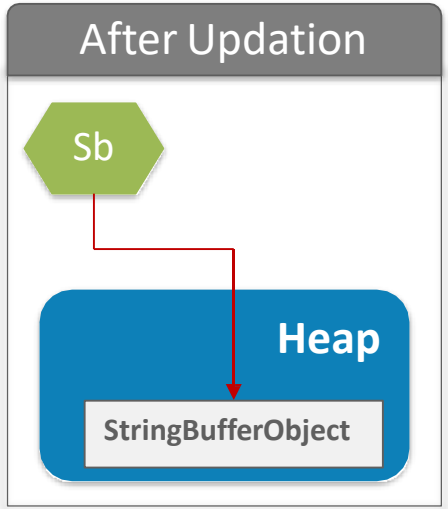
Converts the given string to uppercase

trim()

Removes the leading and trailing spaces of the string

String Buffer





- String Buffer is also a Java class to store a string
- String Buffer is mutable, that means the same StringBuffer object can be modified without new memory location

String is read only and immutable : This means, once the string object is defined and assigned a value, it cant be modified in the same memory location. If it is modified, then a new memory location is allocated for the new string object

In Class Question

1. When should we use String and StringBuffer? why?

In Class Question - Solution

1. When should we use String and StringBuffer? why?

Solution: When we do not need to modify string objects then String can be used else using StringBuffer is a better option since it is mutable.

Program on String Buffer

```
public class StringBufferDemo
{

public static void main (String[] args)
  {
   StringBuffer str1 = new StringBuffer ("Hello");
   StringBuffer str2 = new StringBuffer ("World");

   str1.append(str2);
   System.out.println("Appended string is : "+str1);
 }

}
```

Program on String Buffer

```
public class StringBufferDemo
{

public static void main (String[] args)
  {
   StringBuffer str1 = new StringBuffer ("Hello");
   StringBuffer str2 = new StringBuffer ("World");

   str1.append(str2);
   System.out.println("Appended string is : "+str1);
 }

}
```

Out of the program is

Appended string is: HelloWorld

String Builder

- String Builder is same as String Buffer except that String Builder is not thread safe(not Synchronised)
- That means data is not safe when multiple threads run at the same time

For Example: The following code:

```
StringBuilder sb = new StringBuilder(); // creates empty builder, capacity 16

sb.append("Greetings"); // adds 9 character string at beginning
```

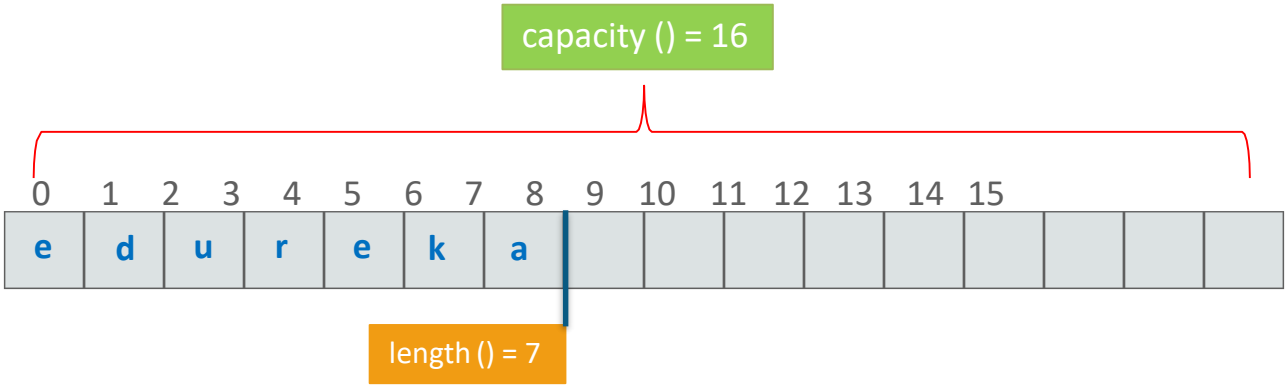
String Builder

- String Builder is same as String Buffer except that String Builder is not thread safe(not Synchronised)
- That means data is not safe when multiple threads run at the same time

For Example: The following code:

```
StringBuilder sb = new StringBuilder(); // creates empty builder, capacity 16

sb.append("Greetings"); // adds 9 character string at beginning
```

will produce a string builder with a length of 9 and a capacity of 16. String builder with a length of 7 and a capacity of 16 is shown below:



String Builder Constructors

StringBuilder Constructors	
Constructor	**Description**
StringBuilder()	Creates an empty string builder with a capacity of 16 (16 empty elements).
StringBuilder(CharSequence cs)	Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence.
StringBuilder(int initCapacity)	Creates an empty string builder with the specified initial capacity.
StringBuilder(String s)	Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string.

Assignment – Single Dimensional Array

- Program to find the largest element in the array
- Program to search an element in the array. Take the data from the user using scanner class
- Program to sort an array in the ascending order
- Program to display second largest element in the array with and without sorting the array
- Program to reverse the elements of the array

Assignment – Two Dimensional Array

- Program to add, subtract two dimensional array of 2 x 2 matrix
- Program to transpose a matrix
- Program to add all the elements of the matrix

Assignment – Functions

- Write a program to find the cube of a number
- Write a program to find factorial of a number
- Write a program to reverse the digits of a number
- Write a program to check given string is palindrome or not
- Write a program to check given number is even or odd
- Write a program to generate 'n' Fibonacci numbers
- Write a program to check given number is prime or not
- Write a program to swap two numbers

Assignment – Functions Overloading

- Write a program to cube of an integer and double values using function overloading
- Write a program to print integer and String data using function overloading

Assignment

1. Write a program to accept 5 employee ids and the corresponding names and their salaries from the user and store them in three arrays. Pass these arrays to a function display() as arguments. This display() will display the content of the arrays in the following format.

ID	Name	Salary
00	Divya	600000
002	Shresta	550000
003	Charan	500000
004	Vineet	500000
005	Jabbar	300000

Assignment (Contd.)

2. Write another function display() with Employee ID array and Employee name array as arguments. (Note: here we are using concept of function overloading). This function will display the content of the 2 arrays in the below given format.

ID	Name
00	Divya
002	Shresta
003	Charan
004	Vineet
005	Jabbar

Assignment (Contd.)

3. Write another function named display() which takes 4 arguments. The arguments are name as String and 3 arrays (Employee id, name and salary). Function prototype looks as given below:

display (String name, int regno[], String Empname[], double salary[]).

This function will search for the name in the Empname array and will display its corresponding id and salary in the below given format. For example, if Divya is given as the name to search then display () function will display the following record.

ID	Name	Salary
00	Divya	600000

Note: main() should have the following steps-

1. Declaring the arrays.
2. Accepting data for the arrays.
3. Calling the 2 display() functions which takes 3 and 2 arguments.
4. Accept a user name to search in the array and display the record by calling the display() function which takes 4 arguments.

Thank You!