

Core JAVA training - index

Date:05/08/2024

1. Language And Applications
2. JAVA Features
 - Why Java is Independent?
 - Oops
 - Exception Handling
 - Multi threading
 - Web Application
 - Open Source
 - Security
 - Support Networking
 - Memory Management
3. JDK,JRE,JVM
4. Basic Java Programming
5. Packages

Date:06/08/2024

Morning(11:00 am)

1. Nested Loops
2. One Dimensional Array
3. Two Dimensional Array
4. Logical Programming

After Noon(3:30 pm)

1. SwitchCase
2. Scanner Class
3. Java.lang
 - Object Class Methods
4. Enum
5. Event Management Application

Date:07/08/2024
Mrng(11:00 am)

1.oops

- Encapsulation
Programs
Calculation
Person
Method Flow

After Noon(3:30 pm)

1. Inheritance
2. Polymorphism
 - Method overloading
 - Method Overriding
3. Abstraction
4. IS-A (Inheritance)
5. HAS- A (Object Creation).

Date:08/08/24

Constructor

- i. Class name and constructor name should be same
- ii. There are 2 types of constructors
 - a. Default Constructor
 - b. Parameterized Constructor
- iii. We can access constructor while creation of object
- iv. Constructors are mainly for initializing
- v. Constructor doesn't have any return type not even void. If you declare as a void the compiler will consider as a method not a constructor
- vi. Every class needs atleast 1 default constructor
- vii. this, super This
 - > this is a keyword always refers to instance variables
- viii. Always constructor are overloaded

Program1:

```
1 package com.evergent.corejava.constructor;
2
3 public class Employee1 {
4
5     public Employee1()
6     {
7         System.out.println("Default constructor..");
8     }
9     public static void main(String[] args) {
10         new Employee1();
11     }
12 }
13
```

Problems @ Javadoc Declaration Console ×

<terminated> Employee1 [Java Application] C:\Users\Shivani.Jakkula\Desktop\eclipse-2023-03\edi
Default constructor..

Program2:

```
1 package com.evergent.corejava.constructor;
2
3 public class Employee2 {
4     int eno;
5     String ename;
6     double sal;
7     public Employee2()
8     {
9         System.out.println("default constructor...");
10    }
11    public Employee2(int eno1,String ename1,double sal1)
12    {
13        eno=enol;
14        ename=ename1;
15        sal=sall;
16    }
17    public void display()
18    {
19        System.out.println("employee no.:"+eno);
20        System.out.println("employee name.:"+ename);
21        System.out.println("employee sal:"+sal);
22    }
23    public static void main(String[] args) {
24        Employee2 emp1=new Employee2();
25        Employee2 emp2=new Employee2(123,"Shiva",60000);
26        emp1.display();
27        emp2.display();
28    }
29 }
30 }
31 }
32 }
```

<terminated> Employee2 [Java Application] C
default constructor...
employee no.:0
employee name.:null
employee sal:0.0
employee no.:123
employee name.:Shiva
employee sal:60000.0

Program3:

```

package com.evergent.corejava.constructor;

public class Employee3 {
    int eno;
    String ename;
    double sal;
    public Employee3()
    {
        System.out.println("default constructor...");
    }
    public Employee3(int eno,String ename,double sal)
    {
        this.eno=eno;
        this.ename=ename;
        this.sal=sal;
    }
    public void display()
    {
        System.out.println("employee no.:"+eno);
        System.out.println("employee name.:"+ename);
        System.out.println("employee sal:"+sal);
    }
    public static void main(String[] args) {
        Employee3 emp1=new Employee3();
        Employee3 emp2=new Employee3(123,"Shiva",60000);
        emp1.display();
        emp2.display();
    }
}

```

```

<terminated> Employee2 [Java Appli
default constructor...
employee no.:0
employee name.:null
employee sal:0.0
employee no.:123
employee name.:Shiva
employee sal:60000.0

```

Program4:

```

package com.evergent.corejava.constructor;

public class Employee4 {

    void Employee4()
    {
        System.out.println("Default constructor..");
    }
    public static void main(String[] args) {
        Employee4 emp4=new Employee4();
        emp4.Employee4();
    }
}

```

```

<terminated> Employee4 [Java Applic
Default constructor..

```

Program5:

```

package com.evergent.corejava.constructor;

public class Employee5 {
    int eno;
    String ename;
    double sal;
    public Employee5()
    {
        System.out.println("default constructor...");
    }
    public Employee5(int eno)
    {
        this.eno=eno;
    }
    public Employee5(int eno,String ename,double sal)
    {
        this(enno);
        this.ename=ename;
        this.sal=sal;
    }
    public void display()
    {
        System.out.println("employee no.:"+eno);
        System.out.println("employee name.:"+ename);
        System.out.println("employee sal:"+sal);
    }
    public static void main(String[] args) {
        Employee5 emp1=new Employee5();
        Employee5 emp2=new Employee5(123,"Shiva",60000);
        emp1.display();
        emp2.display();
    }
}

```

```

<terminated> Employee5 [Java Application] (
default constructor...
employee no.:0
employee name.:null
employee sal:0.0
employee no.:123
employee name.:Shiva
employee sal:60000.0

```

Program 6:

```

class MyEmployee
{
    int eno;
    public MyEmployee()
    {
    }
    public MyEmployee(int eno) {
        System.out.println("My employee no.:"+eno);
    }
}
public class Employee6 extends MyEmployee{
    String name;
    double sal;
    Employee6()
    {
        System.out.println("default constructor..");
    }
    Employee6(int eno,String name,double sal)
    {
        super(enno);
        this.eno=eno;
        this.name=name;
        this.sal=sal;
    }
    public void display()
    {
        System.out.println("employee no.:"+eno);
        System.out.println("employee name.:"+name);
        System.out.println("employee sal:"+sal);
    }
    public static void main(String[] args) {
        Employee6 emp1=new Employee6();
        Employee6 emp2=new Employee6(123,"Shiva",60000);
        emp1.display();
        emp2.display();
    }
}

```

```

<terminated> Employee6 [Java Application] (
default constructor..
My employee no.:123
employee no.:0
employee name.:null
employee sal:0.0
employee no.:123
employee name.:Shiva
employee sal:60000.0

```

Program 7:

```
1 package com.evergent.corejava.constructor;
2
3 public class Car7 {
4     String color;
5     int maxSpeed;
6     Car7()
7     {
8         color="White";
9         maxSpeed=120;
10    }
11    Car7(String color,int maxSpeed)
12    {
13        this.color=color;
14        this.maxSpeed=maxSpeed;
15    }
16    void display()
17    {
18        System.out.println("color:"+color);
19        System.out.println("maxSpeed:"+maxSpeed);
20    }
21    public static void main(String[] args) {
22        Car7 c1=new Car7();
23        Car7 c2=new Car7("red",150);
24        c1.display();
25        c2.display();
26    }
27 }
28 }
```

```
<terminated> Car7 [Java Applicat
color:White
maxSpeed:120
color:red
maxSpeed:150
```

Program8:

```

class Animal
{
    private String name;
    private int age;
    public Animal(String name,int age)
    {
        this.name=name;
        this.age=age;
    }
    public void displayInfo() {
        System.out.println("Name:"+name);
        System.out.println("Age:"+age);
    }
}
class Dog extends Animal{
    private String breed;
    public Dog(String name,int age,String breed)
    {
        super(name,age);
        this.breed=breed;
    }
    public void displayInfo()
    {
        super.displayInfo();
        System.out.println("Breed:"+breed);
    }
}
public class Inheritance_Overriding8 {
    public static void main(String[] args) {
        Dog dog=new Dog("Buddy",5,"Golden Retriever");
        dog.displayInfo();
    }
}

```

```

<terminated> Inheritance_Overriding8.p
Name:Buddy
Age:5
Breed:Golden Retriever

```

Program9:

```
package com.evergent.core.java.constructor;
```

```
public class Student9 {  
    String name;  
    int age;  
    public Student9(String name,int age)  
    {  
        this.name=name;  
        this.age=age;  
    }  
    public Student9(Student9 s)  
    {  
        this.name=s.name;  
        this.age=s.age;  
    }  
    public void displayDetails()  
    {  
        System.out.println("Name:"+name);  
        System.out.println("Age:"+age);  
    }  
    public static void main(String[] args) {  
        Student9 student1=new Student9("shiva",20);  
        Student9 student2=new Student9(student1);  
        student1.displayDetails();  
        student2.displayDetails();  
    }  
}
```

```
<terminated> Student9.java  
Name:shiva  
Age:20  
Name:shiva  
Age:20
```


Date:09/08/2024 - Day5

1. Static

- a. Static is a keyword
- b. We can declare variables and methods as static
- c. We can access static variables and static methods directly through classname.methodname and classname.variablename respectively.
- d. Static methods can access static methods and static variables only.
- e. Static methods cannot access non static methods and non static variables.
- f. Non static methods can access static methods and static variables.
- g. Static block- whenever class is loaded inside the JVM at that time static block is initiated.

Program1:

```
1 package com.evergent.corejava.staticprograms;
2
3 public class StaticDemo1 {
4     static String cname="India";
5     static public void myData()
6     {
7         System.out.println("static method:myData");
8     }
9     public static void main(String[] args) {
10        System.out.println(cname);
11        myData();
12    }
13 }
```

```
<terminated> StaticDemo1 [Java Application] C:\Use
India
static method:myData
```

Program2:

```
1 package com.evergent.corejava.staticprograms;
2
3 public class StaticDemo2 {
4     static String cname="India";
5     static public void myData()
6     {
7         System.out.println("static method:myData");
8     }
9     public static void main(String[] args) {
10        System.out.println(StaticDemo2.cname);
11        StaticDemo2.myData();
12    }
13 }
```

```
<terminated> StaticDemo2 [Java
India
static method:myData
```

Program3:

```

package com.evergent.corejava.staticprograms;

public class StaticDemo3 {
    static String cname="India";
    String name="shiva";
    static public void myData()
    {
        System.out.println("my data..");
        //myShow(); static method can't access non static method
    }
    public void myShow()
    {
        System.out.println("my show");
    }
    public static void main(String[] args) {
        //name;          Cannot make a static reference to the non-static field name
        myData();
    }
}

```

<terminated> StaticDemo3 [J...
my data..

Program4:

```

package com.evergent.corejava.staticprograms;

public class StaticDemo4 {
    static String cName="India";
    String name="Shiva";
    static public void myData()
    {
        System.out.println("my data:"+cName);
    }
    public void myShow()
    {
        myData();
        System.out.println(cName);
    }
    public static void main(String[] args) {
        myData();
        System.out.println(cName);
        StaticDemo4 s4=new StaticDemo4();
        s4.myShow();
    }
}

```

<terminated> StaticDemo4 [Java...
my data:India
India
my data:India
India

Program5:

```

package com.evergent.corejava.staticprograms;

public class StaticDemo5 {
    static
    {
        System.out.println("static block:open db
    }
    static String cname="India";
    static public void myData()
    {
        System.out.println("my data");
    }
    public static void main(String[] args) {
        System.out.println(StaticDemo5.cname);
        StaticDemo5.myData();
    }
}

```

```

<terminated> StaticDemo5 [Java Application] C:\Users\Shivani.Ji
static block:open db/network connection
India
my data

```

Program6:

```

package com.evergent.corejava.staticprograms;
//if we modify static variable it reflected glo
public class Person6 {
    static String name="shiva";
    int age=22;
    String address="Hyderabad";
    public void display()
    {
        name="welcome";
        System.out.println("Name:"+name);
        System.out.println("Age:"+age);
        System.out.println("Address:"+address);
    }
    public static void main(String[] args) {
        Person6 p1=new Person6();
        //System.out.println(name);
        p1.display();
        //System.out.println(name);
        Person6 p2=new Person6();
        System.out.println(name);
    }
}

```

```

<terminated> Person6 [Java
Name:welcome
Age:22
Address:Hyderabad
welcome

```

2. Final

- Final is a Keyword.
- We can declare a variable, method, or a class as final.
- Final variable cannot be modified.
- Final Method cannot be overridden.
- Final class cannot be inherited.

Program1:

```
package com.evergent.corejava.finalprograms;
```

```
public class FinalDemo1 {  
    final String cname="India";  
    public void myData()  
    {  
        //cname="welcome"; final cname cannot  
        System.out.println(cname);  
    }  
    public static void main(String[] args) {  
        FinalDemo1 fd1=new FinalDemo1();  
        fd1.myData();  
    }  
}
```

<terminated> FinalDer
India

Program2:

```
package com.evergent.corejava.finalprograms;  
  
class MyClass{  
    final public void myProducts()  
    {  
        System.out.println("AI products..");  
    }  
}  
public class FinalDemo2 extends MyClass {  
  
    final String cname="India";  
    /*Cannot override the final method from MyCl  
    public void myProducts()  
    {  
        System.out.println("AI products..");  
    }*/  
    public void myData()  
    {  
        System.out.println(cname);  
    }  
    public static void main(String[] args) {  
        FinalDemo2 fd1=new FinalDemo2();  
        fd1.myData();  
    }  
}
```

<terminated> Fir
India

Program3:

```
package com.evergent.corejava.finalprograms;

final class MyClass1{
    final public void myProducts()
    {
        System.out.println("AI products..");
    }
}
//The type FinalDemo3 cannot subclass the final
public class FinalDemo3 {

    final String cname="India";
    /*Cannot override the final method from MyCl
    public void myProducts()
    {
        System.out.println("AI products..");
    }*/
    public void myData()
    {
        System.out.println(cname);
    }
    public static void main(String[] args) {
        FinalDemo3 fd1=new FinalDemo3();
        fd1.myData();
        MyClass1 m1=new MyClass1();
        m1.myProducts();

    }

}
```

<terminated> FinalDemo3

India

AI products..

12/08/24 :

Strings:

-Why string is immutable?

- String is a final class
- Strings are immutable
- Strings having methods
- All methods are non-synchronized

StringBuffer:

- String buffer is a final class

- b. String buffer is mutable
- c. String buffer having methods
- d. All methods are synchronized

String Builder:

- a. String builder is a final class
- b. String builder is mutable
- c. String builder having methods
- d. All methods are non-synchronized

```
package com.evergent.corejava.strings;

public class StringDemo1 {

    public static void main(String[] args) {
        String str1=new String("Java");
        String str2=new String("Java");
        if(str1==str2)
        {
            System.out.println("True");
        }
        else
        {
            System.out.println("False");
        }
    }
}
```

<terminated> StringDemo1 [Java Application] C:\Users\sandhya.pupp
False

```
package com.evergent.corejava.strings;

public class StringDemo2 {

    public static void main(String[] args) {
        String s1="Java";
        String s2="Java";
        if(s1==s2)
        {
            System.out.println("True");
        }
        else
        {
            System.out.println("Flase");
        }
    }
}
```

<terminated> StringDemo1 [Java Application] C:\Users\sandhya.pur
False

```
package com.evergent.corejava.strings;
//String methods

public class stringDemo3_methods {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String name=new String("    JAVAworld");
        System.out.println("length:"+name.length());
        System.out.println("lowercase:"+name.toLowerCase());
        System.out.println("uppercase:"+name.toUpperCase());
        System.out.println("trim:"+name.trim());
        // trim removes spaces before and after the string
    }
}
```

<terminated> stringDemo3_methods [Java Application]
length:16
lowercase: javaworld
uppercase: JAVAWORLD
trim:JAVAworld

1.create a java program that creates a string and checks if it contains specific subString and then prints out the result

```
package com.evergent.corejava.strings;

public class ContainsSubstring {

    public static void main(String[] args) {
        String str="The Quick brown oooofox jumps over the lazy dog";
        String substr="fox";
        boolean contains=str.contains(substr);
        System.out.println("contains " + substr + " : "+contains);
    }
}
```

<terminated> ContainsSubstring [Java Application] C:\Users
contains fox : true

2. Write a java prgm to create a String ,remove all spaces from the string and then print out the result

```
//Write a java Program to create a string,
package com.evergent.corejava.strings;

public class RemovesSpaces {

    public static void main(String[] args) {
        String str="Hello worls,this is a test";
        String nospaces=str.replace(" ","");
        System.out.println(nospaces);
    }
}
```

```
<terminated> RemovesSpaces [Java Application] C:\Users\san...
Helloworls,thisisatest
```

String concatenation:

Strings can be concatenated using + operator (or) concat.

```
//Write a java program to concate the string.
package com.evergent.corejava.strings;

public class String_Concat {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String str="Hello";
        str=str.concat(" World");
        System.out.println("concatinated string:"+str);
    }
}
```

```
<terminated> String_Concat [Java Application] C:\Users\sanc...
concatinated string:Hello World
```

Reverse of a String:

```
//Java program to reverse a string.
package com.evergent.corejava.strings;

public class ReverseString {

    public static void main(String[] args) {
        String str="Hello World";
        StringBuilder sb=new StringBuilder(str).reverse();
        System.out.println("reversed string is:"+sb);
    }
}
```

```
<terminated> ReverseString [Java Application] C:\Users\sanc...
reversed string is:dlrow olleH
```

Splitting of any sentence (or) text on spaces

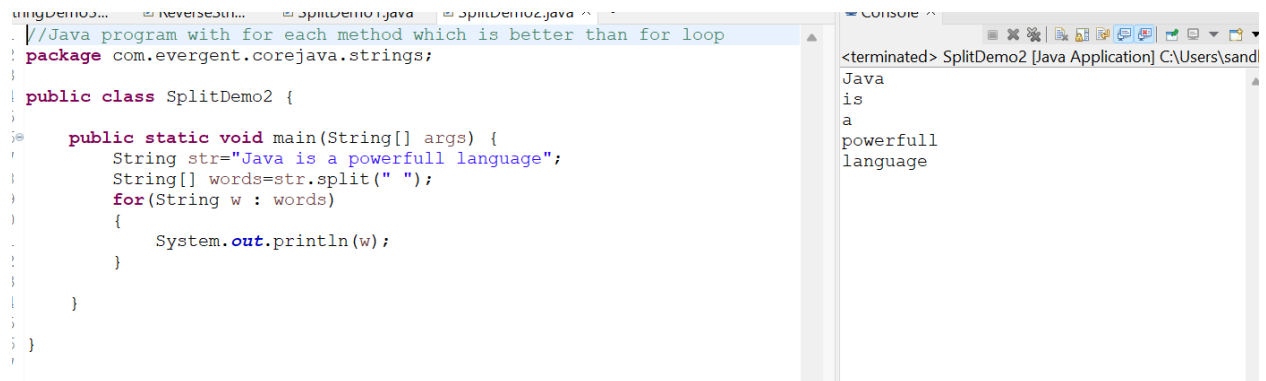
```
package com.evergent.corejava.strings;

public class SplitDemo1 {

    public static void main(String[] args) {
        String str="Java is a powerful language";
        String[] words=str.split(" ");
        for(int i=0;i<words.length;i++)
        {
            System.out.println(words[i]);
        }
    }
}
```

```
<terminated> SplitDemo1 [Java Application] C:\Users\sandh...
Java
is
a
powerful
language
```


Using for each loop:



```
//Java program with for each method which is better than for loop
package com.evergent.corejava.strings;

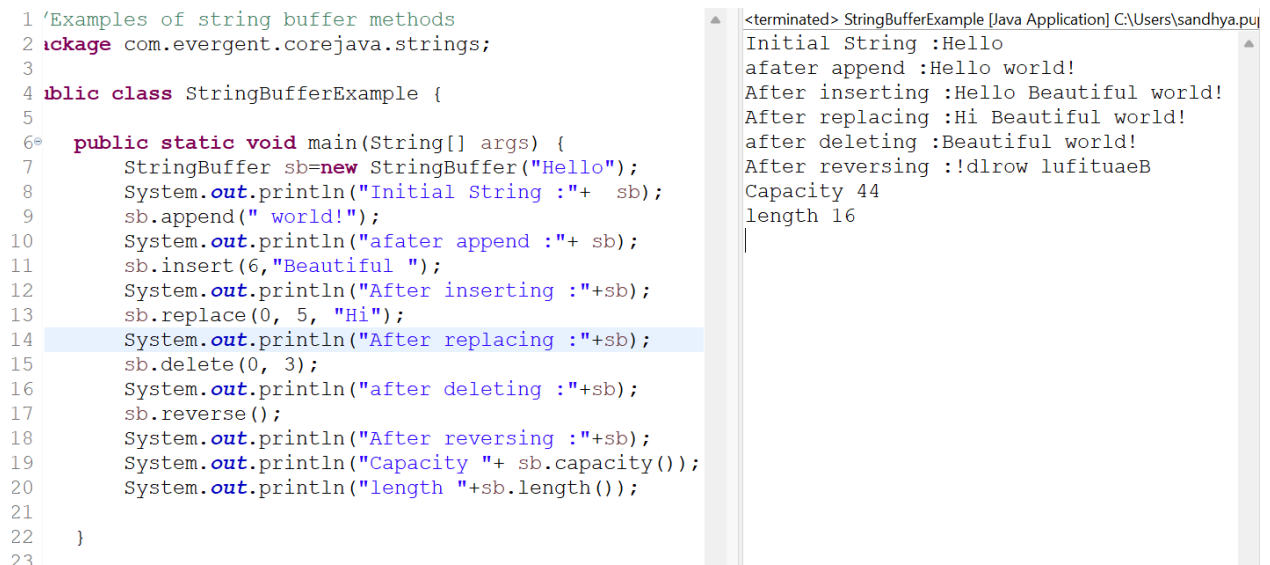
public class SplitDemo2 {

    public static void main(String[] args) {
        String str="Java is a powerfull language";
        String[] words=str.split(" ");
        for(String w : words)
        {
            System.out.println(w);
        }
    }
}
```

Console output:

```
<terminated> SplitDemo2 [Java Application] C:\Users\sandh...
Java
is
a
powerfull
language
```

String Buffer Examples:

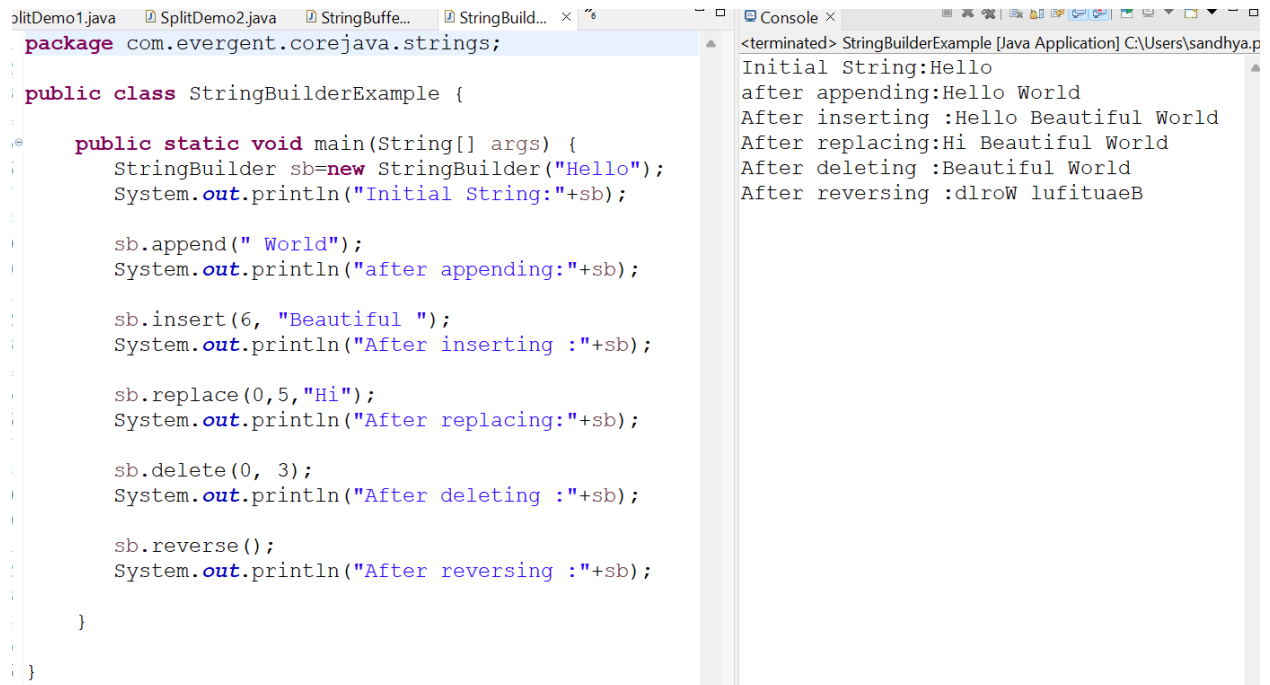


```
1 /Examples of string buffer methods
2 package com.evergent.corejava.strings;
3
4 public class StringBufferExample {
5
6     public static void main(String[] args) {
7         StringBuffer sb=new StringBuffer("Hello");
8         System.out.println("Initial String :"+ sb);
9         sb.append(" world!");
10        System.out.println("afater append :"+ sb);
11        sb.insert(6, "Beautiful ");
12        System.out.println("After inserting :"+sb);
13        sb.replace(0, 5, "Hi");
14        System.out.println("After replacing :"+sb);
15        sb.delete(0, 3);
16        System.out.println("after deleting :"+sb);
17        sb.reverse();
18        System.out.println("After reversing :"+sb);
19        System.out.println("Capacity "+ sb.capacity());
20        System.out.println("length "+sb.length());
21    }
22 }
23
```

Console output:

```
<terminated> StringBufferExample [Java Application] C:\Users\sandhya.pu
Initial String :Hello
afater append :Hello world!
After inserting :Hello Beautiful world!
After replacing :Hi Beautiful world!
after deleting :Beautiful world!
After reversing :!dlrow lufituaeB
Capacity 44
length 16
```

StringBuilder Examples



The screenshot shows an IDE with two tabs: 'StringBuild...' and 'Console'. The code in the left tab is a Java program that demonstrates the functionality of the `StringBuilder` class. It starts with a package declaration, then a class `StringBuilderExample` containing a `main` method. The `main` method creates a `StringBuilder` object with the initial string "Hello". It then performs several operations: appending " World", inserting "Beautiful " at index 6, replacing the first 5 characters with "Hi", deleting the first 3 characters, and finally reversing the entire string. Each operation is followed by a print statement showing the current state of the `StringBuilder`. The console on the right shows the output of these operations, confirming the state of the string after each step.

```
package com.evergent.corejava.strings;

public class StringBuilderExample {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder("Hello");
        System.out.println("Initial String:"+sb);

        sb.append(" World");
        System.out.println("after appending:"+sb);

        sb.insert(6, "Beautiful ");
        System.out.println("After inserting :"+sb);

        sb.replace(0,5,"Hi");
        System.out.println("After replacing:"+sb);

        sb.delete(0, 3);
        System.out.println("After deleting :"+sb);

        sb.reverse();
        System.out.println("After reversing :"+sb);
    }
}
```

<terminated> StringBuilderExample [Java Application] C:\Users\sandhya.p
Initial String:Hello
after appending:Hello World
After inserting :Hello Beautiful World
After replacing:Hi Beautiful World
After deleting :Beautiful World
After reversing :dlroW lufituaeB

String class important points:

1. In java a string is a sequence of characters ,often used to represent text.
2. Strings are objects in java and are instances of the string class,which is part of the java java.lang package
3. Key features of strings in java:
 - A. Immutable:once a string object is created ,it cannot be changed .
4. Java optimizes memory usage by storing strings in special area of memory as string pool
5. If two strings have the same value and are created without using new keyword they will refer to same object in the stringpool.
6. We can create a string in java in multiple ways:
 - a. Using string literals :str="hello world";
 - b. Using the new keyword
String str=new String("hello, world");

String Performance Examples:

1)

```
1 package com.evergent.corejava.strings;
2
3 public class StringPerformance1 {
4
5     public static void main(String[] args) {
6         String a,b;
7         System.out.println('a'+b);
8         System.out.println('a'+3);
9     }
10 }
11
12 }
13
```

<terminated> StringPerformance1 [Java Application] C:\Users\sandhya.pu
195
100

2)

```
package com.evergent.corejava.strings;

public class StringPerformance2 {

    public static void main(String[] args) {
        System.out.println((char) ('a'+3));
        System.out.println("a"+"b");
    }

}
```

<terminated> StringPerformance2 [Java Application] C:\Users\sandhya.pu
d
ab

3)

```
package com.evergent.corejava.strings;

public class StringPerformance3 {

    public static void main(String[] args) {
        String series="";
        for(int i=0;i<26;i++)
        {
            char ch=(char) ('a'+i);
            series+=ch;
        }
        System.out.println(series);
    }

}
```

<terminated> StringPerformance3 [Java Application] C:\Users\sandhya.pu
abcdefghijklmnopqrstuvwxyz

4)

```

package com.evergent.corejava.strings;

public class StringPerformance4 {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder()
        for(int i=0;i<26;i++)
        {
            char ch=(char) ('a'+i);
            sb.append(ch);

        }
        System.out.println(sb);
        // System.out.println(sb.toString());
    }
}

```

```

<terminated> StringPerformance4 [Java Application] C:\Users\sandhya.pup
abcdefghijklmnopqrstuvwxyz

```

5)

```

package com.evergent.corejava.strings;
import java.util.*;

public class StringPerformance5 {

    public static void main(String[] args) {
        String name="JavaTechnologies";
        String str=Arrays.toString(name.toCharArray());
        System.out.println(str);
        System.out.println(name.indexOf('T'));
        System.out.println("    JAVA    ".strip());
    }
}

```

```

<terminated> StringPerformance5 [Java Application] C:\Users\sandhya.puppal
[J, a, v, a, T, e, c, h, n, o, l, o, g, i, e, s]
4
JAVA

```

Interface

- 1) Interface is a Keyword.
- 2) We can declare methods signatures only, but not implementation.
- 3) By default, all interface methods are abstract.
- 4) If any class implements interface the class should override all interface methods otherwise the class will be showing compile time error.
- 5) We cannot create object to interface but we can create reference to interface.

- 6) We can declare variables inside interface all are public static final.
- 7) Java will support multiple inheritance through interface.
- 8) One class can implements more than one interface.
- 9) One interface can extend other interface.

Program for first 4 and 6th points

```
Book.java × BookImpl1.java × BookImpl2.java × NewBook.java × MyNewData.java
package com.evergent.corejava.interfacel;

public interface Book {
    String cName="India";
    abstract public void bookTitle();
    public void bookAuthor();
    public void bookPrice();
}

Book.java × BookImpl1.java × BookImpl2.java × StringPerfo... × Console ×
package com.evergent.corejava.interfacel;

public class BookImpl1 implements Book{
    public void bookTitle()
    {
        System.out.println("Core Java");
    }
    public void bookAuthor()
    {
        System.out.println("Oracle crop"+cName);
    }
    public void bookPrice()
    {
        System.out.println("Rs 550");
    }
    public void show()
    {
        System.out.println("Local Methods of BookImpl");
    }
    public static void main(String args[])
    {
        BookImpl1 book=new BookImpl1();
        book.bookAuthor();
        book.bookTitle();
        book.bookPrice();
        book.show();
    }
}
```

<terminated> BookImpl1 [Java Application] C:\Users\sandhye
Oracle cropIndia
Core Java
Rs 550
Local Methods of BookImpl

2) program for 5th point

We cannot create object to interface but we can create reference to interface.

```

1 package com.evergent.corejava.interface1;
2
3 public interface Book {
4     String cName="India";
5     abstract public void bookTitle();
6     public void bookAuthor();
7     public void bookPrice();
8 }
9
10

```

The screenshot shows an IDE with two tabs: BookImpl1.java and BookImpl2.java. The BookImpl2.java tab is active, displaying the following code:

```

1 //here we cannot call the local methods
2 package com.evergent.corejava.interface1;
3
4 public class BookImpl2 implements Book{
5     public void bookTitle()
6     {
7         System.out.println("Core Java");
8     }
9     public void bookAuthor() {
10        System.out.println("Oracle crop:"+cName);
11    }
12    public void bookPrice()
13    {
14        System.out.println("Rs 550");
15    }
16    public void show()
17    {
18        System.out.println("Local methods of BookImpl2");
19    }
20
21    public static void main(String[] args) {
22        Book b2=new BookImpl2();
23        b2.bookAuthor();
24        b2.bookPrice();
25        b2.bookTitle();
26    }
27 }
28
29
30
31

```

The console output on the right shows the execution results:

```

<terminated> BookImpl2 [Java Application] C:\Users\sandhya.puppal
Oracle crop:India
Rs 550
Core Java

```

3) Program for 7th point

Java will support multiple inheritance through interface.

```

Book.java    BookImpl1.java    BookImpl2.java    NewBook.java
1 package com.evergent.corejava.interface1;
2
3 public interface NewBook {
4     public void myNewBook();
5     public void bookPrice();
6 }
7
8

```

```
BookImpl3.java  com.evergent.corejava.interface1  com.evergent.corejava.interface1  com.evergent.corejava.interface1  com.evergent.corejava.interface1
1 package com.evergent.corejava.interface1;
2
3 public class BookImpl3 implements Book, NewBook{
4     public void myNewBook()
5     {
6         System.out.println("My New Book");
7     }
8     public void bookTitle()
9     {
10        System.out.println("Core Java");
11    }
12    public void bookAuthor() {
13        System.out.println("Oracle crop:"+cName);
14    }
15    public void bookPrice()
16    {
17        System.out.println("Rs 550");
18    }
19    public void show()
20    {
21        System.out.println("Local methods of BookImpl1");
22    }
23    // public void dataInfo()
24    // {
25    //     System.out.println("My data Info");
26    // }
27    public static void main(String args[])
28    {
29        BookImpl3 book=new BookImpl3();
30        book.myNewBook();
31        book.bookAuthor();
32        book.bookPrice();
33        book.bookTitle();
34        book.show();
35        // book.dataInfo();
36    }
37 }
```

```
<terminated> BookImpl3 [Java Application] C:\Users\sandhya.puppala\
My New Book
Oracle crop:India
Rs 550
Core Java
Local methods of BookImpl1
```

4) Program for 9th point.

One interface can extend other interface.

```
package com.evergent.corejava.interface1;

public interface MyNewData {
    public void dataInfo();
}
```

```
package com.evergent.corejava.interface1;

public interface NewBook extends MyNewData{
    public void myNewBook();
    public void bookPrice();
}
```

```

package com.evergent.corejava.interface1;

public class BookImpl3 implements Book,NewBook{
    public void myNewBook()
    {
        System.out.println("My New Book");
    }
    public void bookTitle()
    {
        System.out.println("Core Java");
    }
    public void bookAuthor() {
        System.out.println("Oracle crop:"+cName);
    }
    public void bookPrice()
    {
        System.out.println("Rs 550");
    }
    public void show()
    {
        System.out.println("Local methods of BookImpl1");
    }
    public void dataInfo()
    {
        System.out.println("My data Info");
    }
    public static void main(String args[])
    {
        BookImpl3 book=new BookImpl3();
        book.myNewBook();
        book.bookAuthor();
        book.bookPrice();
        book.bookTitle();
        book.show();
        book.dataInfo();
    }
}

```

<terminated> BookImpl3 [Java Application] C:\Users\sandhya.puppala\Des

```

My New Book
Oracle crop:India
Rs 550
Core Java
Local methods of BookImpl1
My data Info

```


Abstract Class

- 1) Abstract is Keyword.
- 2) Abstract class having abstract methods and concrete(implemented) methods.
- 3) If any class having one abstract method,the class should be declare as abstract keyword,otherwise the class will be showing compile time error.
- 4) If any class extends abstract class,the class should be override all abstract methods,Otherwise the class will be showing compile time error.
- 5) We cannot create object to abstract to Abstract class but we can create reference to Abstract class.
- 6) When we extend one abstract either we should over ride the abstract method (or) class should be declared as Abstract.
- 7) Uses of Abstraction is,not implemented methods can be implemented methods should be used straight away.
- 8) We can declare Abstract class also with zero abstract methods,we cannot create object.

Program 1

```
//abstract class
package com.evergent.corejava.abstract1;

abstract public class Product {
    abstract public void newProducts ();
    public void allProducts ()
    {
        System.out.println("All products");
    }
}
```

```
Employee5.java Product.java ProductImpl1... x ProductImpl2...
1 //Implementation class
2 package com.evergent.corejava.abstract1;
3
4 public class ProductImpl1 extends Product {
5     public void newProducts()
6     {
7         //Implementation of abstract method
8         System.out.println("My new Product");
9     }
10    public void show()
11    {
12        System.out.println("Local methods of ProductImpl1");
13    }
14    public static void main(String args[])
15    {
16        ProductImpl1 p1=new ProductImpl1();
17        p1.show();
18        p1.newProducts();
19    }
20 }
21
```

```
<terminated> ProductImpl1 [Java Application] C:\Users\sandhya.puppala\Des
Local methods of ProductImpl1 class
My new Product
```

Program 2)

```
//abstract class
package com.evergent.corejava.abstract1;

abstract public class Product {
    abstract public void newProducts();
    public void allProducts()
    {
        System.out.println("All products");
    }
}
```

```
Employee5.java Product.java ProductImpl1... ProductImpl2... x
1//this code is for telling that we cannot create object to
2//abstract class but we can create the reference
3 package com.evergent.corejava.abstract1;
4
5 public class ProductImpl2 extends Product{
6     public void newProducts()
7     {
8         System.out.println("My new Product");
9     }
10    public void show()
11    {
12        System.out.println("Local methods of productImpl2");
13    }
14    public static void main(String args[])
15    {
16        Product p2=new ProductImpl2();
17        p2.allProducts();
18        p2.newProducts();
19    }
20 }
21
```

```
<terminated> ProductImpl2 [Java Application] C:\Users\sandhya.puppala\Des
All products
My new Product
```

Exception Handling

- 1) Exception Handling is Mechanism.
- 2) Exceptions are built Mechanism of java.
- 3) All Exception are executed while abnormal conditions only.
- 4) If there is Normal flow it wont execute any exceptions.
- 5) Once any exceptions are occurring in java then remaining lines of code is unreachable.
- 6) Java.lang.Throwable is super class for Exception and errors.
- 7) There are 2 types of Exceptions are there in Java
 - A) Checked Exceptions
 - B) Unchecked Exceptions
- 8) All checked Exceptions are compile time exceptions.
- 9) All Unchecked Exceptions are Runtime Exceptions.
- 10) There are 5 keywords in Exception Handling.
 - A) Try
 - B) Catch()
 - C) Finally
 - D) throws
 - E) Throw
- 11) try is for business logic
- 12) Catch is for handling Exceptions
- 13) finally is block,if exceptions id occurs or not,finally block will execute.
- 14) throws an exception will be executed method by method
- 15) throw is for runtime exceptions and will call predicted exceptions or user defined exception.
- 16) try followed by either catch block or finally block.
- 17) We should follow exceptions hierarchial.
- 18) We can create our own (user defined) exceptions
- 19) Our own exceptions extends exception or Runtime exception.
- 20) All exceptions extends exception or runtime exception
- 21) There is two exception in class,Developer should be handle one after one.
- 22) Developer cant handle the error.

Program for first 2 points

Exception Handling is Mechanism.

Exceptions are built Mechanism of java.



```
1 //1)Exception Handling is Mechanism.
2 //2)Exceptions are built Mechanism of java.
3 package com.evergent.corejava.exceptionhandling;
4
5 public class ExceptionDemo1 {
6
7     public static void main(String[] args) {
8         String name="null";
9         System.out.println(name.length());
10    }
11
12 }
```

Console ×

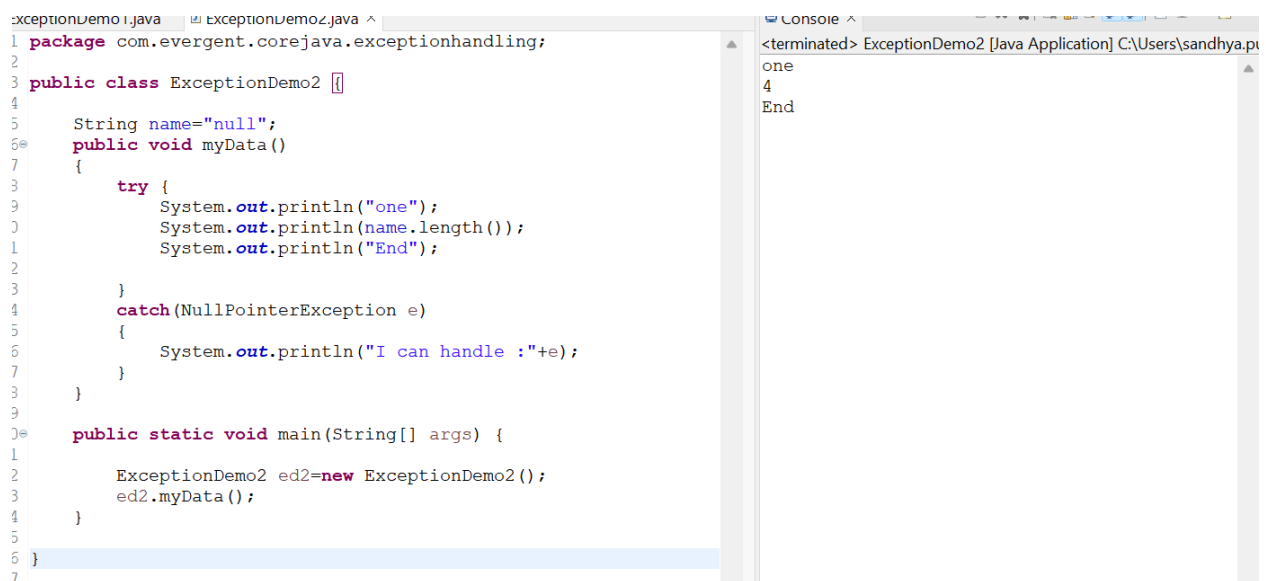
<terminated> ExceptionDemo1 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

Program for 3,4,5 points

All Exception are executed while abnormal conditions only.

If there is Normal flow it wont execute any exceptions.

Once any exceptions are occurring in java then remaining lines of code is unreachable.



```
1 package com.evergent.corejava.exceptionhandling;
2
3 public class ExceptionDemo2 {
4
5     String name="null";
6     public void myData()
7     {
8         try {
9             System.out.println("one");
10            System.out.println(name.length());
11            System.out.println("End");
12        }
13        catch (NullPointerException e)
14        {
15            System.out.println("I can handle :"+e);
16        }
17    }
18
19     public static void main(String[] args) {
20
21         ExceptionDemo2 ed2=new ExceptionDemo2();
22         ed2.myData();
23     }
24 }
```

Console ×

<terminated> ExceptionDemo2 [Java Application] C:\Users\sandhya.puppala\Desktop\

one
4
End

Program for point 21

23) There is two exception in class,Developer should be handle one after one.

```
ExceptionDemo3.java ×
1 //21)There is two exception in class,Developer should be
2 package com.evergent.corejava.exceptionhandling;
3
4 public class ExceptionDemo3 {
5     String name="null";
6     int k=2;
7     public void myData()
8     {
9         try {
10             System.out.println("one");
11             System.out.println(name.length());
12             int t=10/k;
13             System.out.println(t);
14             System.out.println("end");
15         }
16         catch (NullPointerException e)
17         {
18             System.out.println("I can handle "+ e);
19         }
20         catch (ArithmeticException e)
21         {
22             System.out.println("I can handle :"+ e);
23         }
24     }
25
26
27     public static void main(String[] args) {
28         ExceptionDemo3 ed3= new ExceptionDemo3();
29         ed3.myData();
30
31     }
32
33 }
34
```

```
Console ×
<terminated> ExceptionDemo3 [Java Application] C:\Users\sandhya.puppala\Desk
one
4
5
end
```

Program for 17 point

We should follow exceptions hierarchial.

The screenshot shows an IDE with two panes. The left pane displays the source code for `ExceptionDemo4.java`. The code defines a class `ExceptionDemo4` with a `myData()` method that uses a `try-catch` block to handle `NullPointerException`, `ArithmeticException`, and a general `Exception`. The `main` method creates an instance of the class and calls `myData()`. The right pane shows the console output, which displays the results of the program's execution.

```
1 //17)We should follow exceptions hierarchial.  
2 package com.evergent.corejava.exceptionhandling;  
3  
4 public class ExceptionDemo4 {  
5     String name="null";  
6     int k=2;  
7     public void myData()  
8     {  
9         try {  
10             System.out.println("one");  
11             System.out.println(name.length());  
12             int t=10/k;  
13             System.out.println(t);  
14             System.out.println("end");  
15         }  
16         catch (NullPointerException e)  
17         {  
18             System.out.println("I can handle "+ e);  
19         }  
20         catch (ArithmeticException e)  
21         {  
22             System.out.println("I can handle :"+ e);  
23         }  
24         catch (Exception e)  
25         {  
26             System.out.println("I can handle :"+e);  
27         }  
28     }  
29  
30  
31     public static void main(String[] args) {  
32         ExceptionDemo4 ed3= new ExceptionDemo4();  
33         ed3.myData();  
34     }  
35 }  
36  
37 }  
38 }
```

Console Output:

```
<terminated> ExceptionDemo4 [Java Application] C:\Users\sandhya.puppala\Desktop  
one  
4  
5  
end
```

Program for 13 point

finally is block,if exceptions id occurs or not,finally block will execute.

```
ExceptionDemo5.java
1 //13)finally is block,if exceptions id
2 //occurs or not,finally block will execute.
3 package com.evergent.corejava.exceptionhandling;
4
5 public class ExceptionDemo5 {
6     String name="null";
7     int k=2;
8     public void myData()
9     {
10         try {
11             System.out.println("one");
12             System.out.println(name.length());
13             int t=10/k;
14             System.out.println(t);
15             System.out.println("end");
16         }
17         catch(NullPointerException e)
18         {
19             System.out.println("I can handle "+ e);
20         }
21         catch(ArithmeticException e)
22         {
23             System.out.println("I can handle :"+ e);
24         }
25         catch(Exception e)
26         {
27             System.out.println("I can handle :"+e);
28         }
29         finally {
30             System.out.println("finally block close connection ")
31         }
32     }
33
34
35     public static void main(String[] args) {
36         ExceptionDemo5 ed3= new ExceptionDemo5();
37         ed3.myData();
38     }
39 }
40
41 }
```

```
<terminated> ExceptionDemo5 [Java Application] C:\Users\sandhya
one
4
5
end
finally block close connection
```

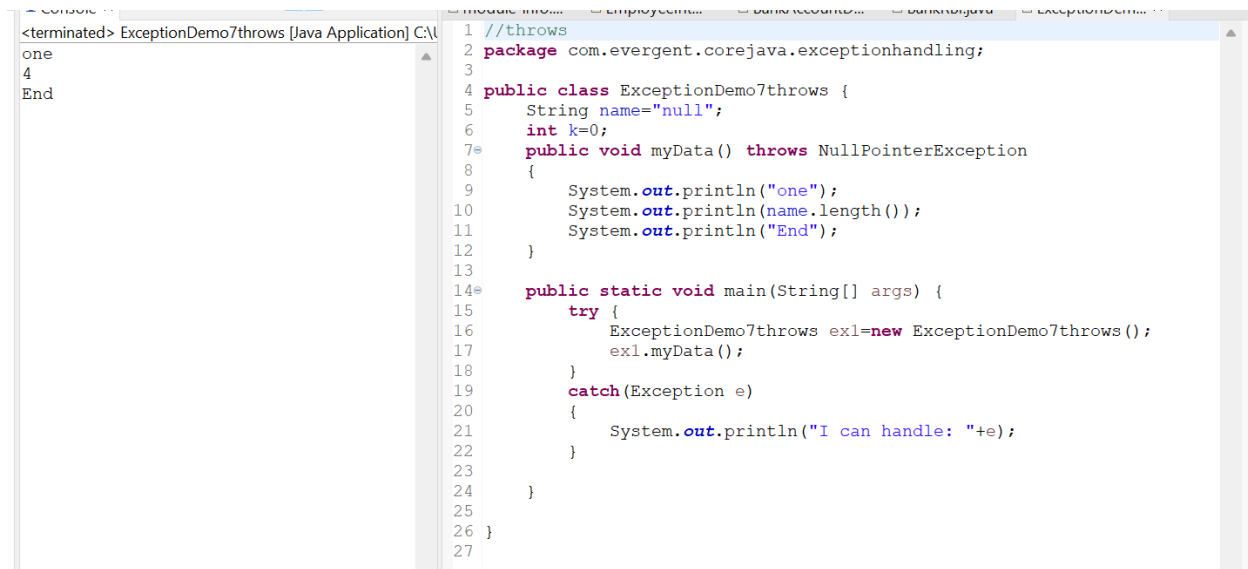
Program for 16 point
try followed by either catch block or finally block.

```
ExceptionDemo6.java x ExceptionDemo5.java
1 //try followed by either catch block or
2 //finally block.
3 package com.evergent.corejava.exceptionhandling;
4
5 public class ExceptionDemo6 {
6
7     String name="null";
8     public void myData()
9     {
10         try {
11             System.out.println("one");
12             System.out.println(name.length());
13             System.out.println("End");
14         }
15         finally {
16             System.out.println("finally block");
17         }
18     }
19
20     public static void main(String[] args) {
21         ExceptionDemo6 ed2=new ExceptionDemo6();
22         ed2.myData();
23     }
24 }
25 }
```

```
<terminated> ExceptionDemo6 [Java Application] C:\Users\sandhya.puppa
one
4
End
finally block
```

Throws:

Program 1



The screenshot shows an IDE with two panels. The left panel is the console, and the right panel is the code editor.

Console Output:

```
<terminated> ExceptionDemo7throws [Java Application] C:\
one
4
End
```

Code Editor:

```
1 //throws
2 package com.evergent.corejava.exceptionhandling;
3
4 public class ExceptionDemo7throws {
5     String name="null";
6     int k=0;
7     public void myData() throws NullPointerException
8     {
9         System.out.println("one");
10        System.out.println(name.length());
11        System.out.println("End");
12    }
13
14    public static void main(String[] args) {
15        try {
16            ExceptionDemo7throws ex1=new ExceptionDemo7throws();
17            ex1.myData();
18        }
19        catch(Exception e)
20        {
21            System.out.println("I can handle: "+e);
22        }
23    }
24 }
25
26 }
27 }
```

Program 2)


```
Product.java  ExceptionDemo8throws.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 public class ExceptionDemo8throws {
4     String name=null;
5     public void myData() throws NullPointerException
6     {
7         System.out.println("one");
8         System.out.println(name.length());
9         System.out.println("end");
10    }
11    public void myChange() throws NullPointerException
12    {
13        myData();
14        System.out.println("Mychange method");
15    }
16
17    public static void main(String[] args) {
18        try {
19            ExceptionDemo8throws ex2=new ExceptionDemo8throws();
20            ex2.myChange();
21        }
22        catch(Exception e)
23        {
24            System.out.println("I can handle :"+e);
25        }
26    }
27
28

```

Console ×

<terminated> ExceptionDemo8throws [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win.x64_2023.03\jre\bin\java.exe
one
I can handle :[java.lang.NullPointerException](#): Cannot invoke "String.length()" because "this.name" is null

Program 3)

User defines exceptions

```
Console ×  ProductImpl3.java ×
<terminated> ProductImpl3 (1) [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\...
Hello :there is no products
com.evergent.corejava.exceptionhandling.ProductNotFoundException

1 package com.evergent.corejava.exceptionhandling;
2 class ProductNotFoundException extends Exception{
3     public ProductNotFoundException(String message)
4     {
5         System.out.println("Hello :"+message);
6     }
7 }
8
9 public class ProductImpl3 {
10     int pno=105;
11     public void myData() throws ProductNotFoundException
12     {
13         if(pno>100)
14         {
15             throw new ProductNotFoundException("there is no product");
16         }
17         else
18         {
19             System.out.println("products are there");
20         }
21     }
22
23     public static void main(String[] args) {
24         try {
25             ProductImpl3 product =new ProductImpl3();
26             product.myData();
27         }
28         catch(ProductNotFoundException e)
29         {
30             System.out.println(e);
31         }
32     }
33 }
34
35
36

```

Program4)

User defined exceptions using super keyword.

```
UserDefinedExceptionDemo10.java ×
1 package com.evergent.corejava.exceptionhandling;
2 class InvalidAgeException extends Exception{
3     public InvalidAgeException(String message)
4     {
5         super(message);
6     }
7 }
8 public class UserDefinedExceptionDemo10 {
9     public static void checkAge(int age) throws InvalidAgeException
10    {
11        if(age<18)
12        {
13            throw new InvalidAgeException("Age must be 18 or older");
14        }
15        else
16        {
17            System.out.println("Access granted-You are old enough");
18        }
19    }
20    public static void main(String[] args) {
21        try {
22            checkAge(16);
23        }
24        catch(InvalidAgeException e)
25        {
26            System.out.println("caught the exception: "+e.getMessage());
27        }
28        System.out.println("Program continues after handling the exception");
29    }
30 }
```

Console ×

```
<terminated> UserDefinedExceptionDemo10 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.c
caught the exception: Age must be 18 or older
Program continues after handling the exception
```

Program 5)

User defined exceptions using super keyword.

```
UserDefinedExceptionDemo11.java ×
1 package com.evergent.corejava.exceptionhandling;
2 class InsufficientFundsException extends Exception{
3     public InsufficientFundsException(String message)
4     {
5         super(message);
6     }
7 }
8 public class UserDefinedExceptionDemo11 {
9     public static void withdraw(double amount) throws InsufficientFundsException
10    {
11        double balance=500.00;
12        if(amount>balance)
13        {
14            throw new InsufficientFundsException("Invalid funds to withdraw");
15        }
16        else
17        {
18            System.out.println("withdraw successful");
19        }
20    }
21    public static void main(String[] args) {
22        try {
23            withdraw(600.00);
24        }
25        catch(InsufficientFundsException e)
26        {
27            System.out.println("caught insufficient InsufficientFundsException: "+e.getMessage());
28            | System.out.println(e);
29        }
30        System.out.println("Program continue after handling the exception");
31    }
32 }
33
```

Console ×

```
<terminated> UserDefinedExceptionDemo11 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.o
caught insufficient InsufficientFundsException: Invalid funds to withdraw
com.evergent.corejava.exceptionhandling.InsufficientFundsException: Invalid funds to withdraw
Program continue after handling the exception
```

Program 6)
Extends the RuntimeException instead of Exception

```
UserDefinedUncheckedExceptionDemo12.java ×
3 {
4     public InvalidScoreException(String message)
5     {
6         super(message);
7     }
8 }
9 public class UserDefinedUncheckedExceptionDemo12 {
10     public static void ValidateScore(int score)
11     {
12         if(score<0 || score>100)
13         {
14             throw new InvalidScoreException("score must be between 0 and 100");
15         }
16         else
17         {
18             System.out.println("score is valid");
19         }
20     }
21     public static void main(String[] args) {
22         try {
23             UserDefinedUncheckedExceptionDemo12 e=new UserDefinedUncheckedExceptionDemo12();
24             e.ValidateScore(110);//we can access without creating the object
25             //because 2 static methods in same class can be accessed without object creation
26         }
27         catch(InvalidScoreException e)
28         {
29             System.out.println("caught the exception: "+e.getMessage());
30             System.out.println(e);
31         }
32         System.out.println("program continues after handling the exception");
33     }
34 }

Console ×
<terminated> UserDefinedUncheckedExceptionDemo12 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.ed
caught the exception: score must be between 0 and 100
com.evergent.corejava.exceptionhandling.InvalidScoreException: score must be between 0 and 100
program continues after handling the exception
```

Program 7)

Program for ArrayIndexOutOfBoundsException

```
ArrayIndexOutOfBounds13.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 public class ArrayIndexOutOfBounds13 {
4
5     public static void main(String[] args) {
6         int[] number= {1,2,3,4,5};
7         try {
8             System.out.println("get the number at index 10:"+number[10]);
9         }
10        catch(ArrayIndexOutOfBoundsException e)
11        {
12            System.out.println("caught at the index:"+e.getMessage());
13        }
14        System.out.println("program continues after the execution");
15    }
16 }
17
18 }
19

Console ×
<terminated> ArrayIndexOutOfBounds13 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openj
caught at the index:Index 10 out of bounds for length 5
program continues after the execution
```

Program 8)

Program for commandlineArguments

```
CommandLineArguments14.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 public class CommandLineArguments14 {
4
5     public static void main(String[] args) {
6         System.out.println(args[0]);
7         System.out.println(args[1]);
8     }
9
10 }
11
12
```

Console ×

```
<terminated> CommandLineArguments14 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk
101
100
```

Program 9)

Example program for FileNotFoundException

```
CommandLineArguments14.java × CompileTimeFileDemo15.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 import java.io.FileNotFoundException;
4 import java.util.Scanner;
5 import java.io.File;
6 public class CompileTimeFileDemo15 {
7     public static void main(String[] args)
8     {
9         try {
10             File file=new File("c:/mydata/myinfo.txt");
11             Scanner sc=new Scanner(file);
12             while(sc.hasNext())
13             {
14                 System.out.println(sc.nextLine());
15             }
16             sc.close();
17         }
18         catch(FileNotFoundException e)
19         {
20             System.out.println("File not found : "+e.getMessage());
21             e.printStackTrace();
22         }
23     }
24 }
25
```

Console ×

```
<terminated> CompileTimeFileDemo15 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk
file for 15 program -sandhya
```

Example program for StackOverflow

```
StackOverflowErrorExample.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 public class StackOverflowErrorExample {
4
5     public static void main(String[] args) {
6         try {
7             recursivemethod();
8         }
9         catch (StackOverflowError e)
10        {
11            System.out.println("stack over flow caught:"+e.getMessage());
12        }
13    }
14
15    public static void recursivemethod() {
16        recursivemethod(); //keeps on calling
17    }
18
19 }
20
```

Console ×

<terminated> StackOverflowErrorExample [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
stack over flow caught:null

Example Program for OutOfMemory

```
StackOverflowErrorExample.java × OutOfMemory.java ×
1 package com.evergent.corejava.exceptionhandling;
2
3 public class OutOfMemory {
4
5     public static void main(String[] args) {
6         Integer[] array=new Integer[1000000000 * 1000000000];
7         System.out.println(array);
8     }
9
10 }
11
12
```

Console ×

<terminated> OutOfMemory [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at corejava_development/com.evergent.corejava.exceptionhandling.OutOfMemory.main(OutOfMemory.java:6)

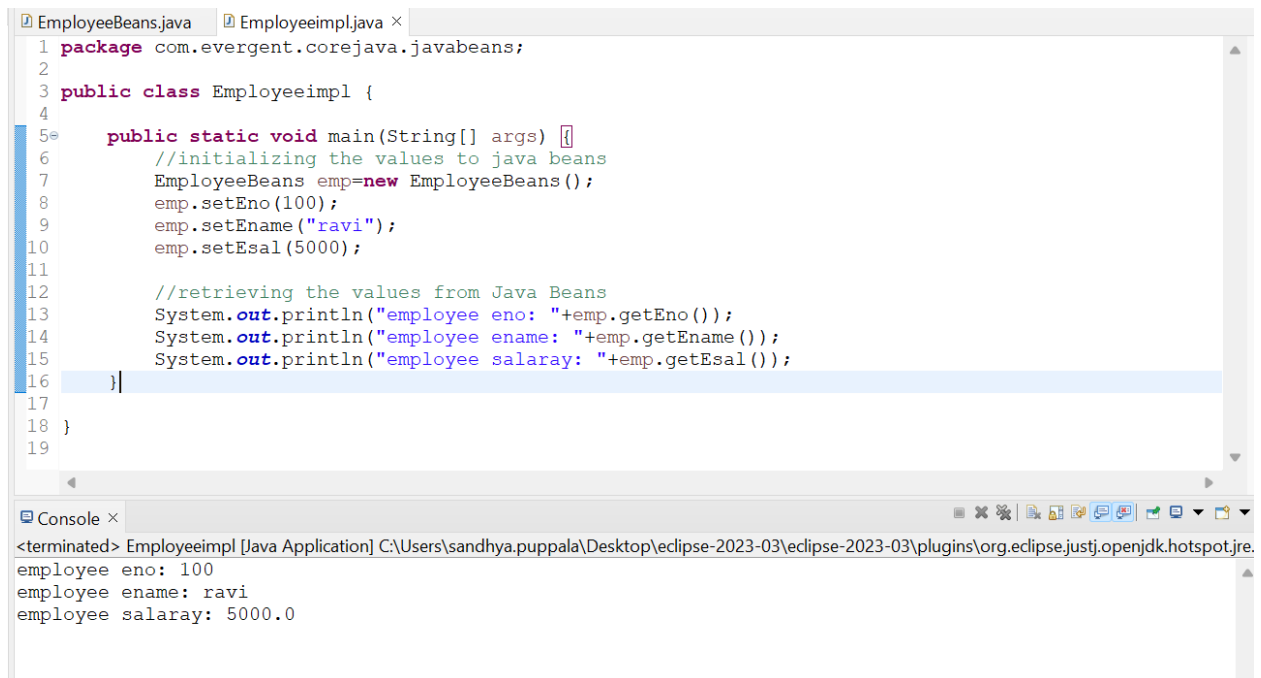
JAVA BEANS

- 1) Java Bean is a mechanism.
- 2) Java Bean is light weight.
- 3) All attributes are private and get/set methods are public.
- 4) Implements java.serializable interface.
- 5) We can achieve tightly encapsulation through Java Beans.

Program 1)

```
EmployeeBeans.java × Employeeimpl.java
1 package com.evergent.corejava.javabeans;
2
3 public class EmployeeBeans {
4     private int eno;
5     private String ename;
6     private double esal;
7     public void setEno(int eno)
8     {
9         this.eno=enno;
10    }
11    public int getEno()
12    {
13        return eno;
14    }
15    public void setName(String ename) {
16        this.ename=ename;
17    }
18    public String getName()
19    {
20        return ename;
21    }
22    public void setEsal(double esal)
23    {
24        this.esal=esal;
25    }
26    public double getEsal() {
27        return esal;
28    }
29
30 }
31
```

Implementation class



The screenshot shows the Eclipse IDE with two tabs: `EmployeeBeans.java` and `Employeeimpl.java`. The `Employeeimpl.java` tab is active, displaying the following code:

```
1 package com.evergent.corejava.javabeans;
2
3 public class Employeeimpl {
4
5     public static void main(String[] args) {
6         //initializing the values to java beans
7         EmployeeBeans emp=new EmployeeBeans();
8         emp.setEno(100);
9         emp.setName("ravi");
10        emp.setEsal(5000);
11
12        //retrieving the values from Java Beans
13        System.out.println("employee eno: "+emp.getEno());
14        System.out.println("employee ename: "+emp.getName());
15        System.out.println("employee salary: "+emp.getEsal());
16    }
17 }
18
19
```

Below the code editor is the Console window, which shows the output of the program:

```
<terminated> Employeeimpl [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.
employee eno: 100
employee ename: ravi
employee salary: 5000.0
```

Program 2)

Program initializing the values with constructor and retrieving the values with java beans

EmployeeBeans.java Employeeimpl.java ProductBeans.java × ProductBeansImpl.java

```
1 package com.evergent.corejava.javabeans;
2
3 public class ProductBeans {
4
5     private int pno;
6     private String pname;
7     private double price;
8     public ProductBeans(int pno,String pname,double price)
9     {
10         this.pno=pno;
11         this.pname=pname;
12         this.price=price;
13     }
14     public int getPno()
15     {
16         return pno;
17     }
18     public String getPname()
19     {
20         return pname;
21     }
22     public double getPrice() {
23         return price;
24     }
25 }
26
```

Implementation class

EmployeeBeans.java Employeeimpl.java ProductBeans.java ProductBeansImpl.java ×

```
1 package com.evergent.corejava.javabeans;
2
3 public class ProductBeansImpl {
4
5     public static void main(String[] args) {
6         ProductBeans p=new ProductBeans(10,"laptop",5000);
7         System.out.println("product pno is: "+p.getPno());
8         System.out.println("product name is: "+p.getPname());
9         System.out.println("product price is: "+p.getPrice());
10
11     }
12
13 }
14
```

Console ×

```
<terminated> ProductBeansImpl [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03
product pno is: 10
product name is: laptop
product price is: 5000.0
```

Program 3)

Retrieving the values with over-riding the toString method

```
EmployeeBeans.java EmployeeImpl.java ProductBeans.java ProductBeansImpl.java StudentBean.java StudentBeansImpl.java
1 package com.evergent.corejava.javabeans;
2
3 public class StudentBean {
4     private int sno;
5     private String sname;
6     private String address;
7     public void setName(String sname)
8     {
9         this.sname=sname;
10    }
11    public void setSno(int sno)
12    {
13        this.sno=sno;
14    }
15    public void setAddress(String address)
16    {
17        this.address=address;
18    }
19    public String toString()
20    {
21        return "student no: " + sno+ "\n student name: " +sname+ "\n student address: "+address;
22    }
23 }
24
```

Implementation class

```
EmployeeBeans.java EmployeeImpl.java ProductBeans.java ProductBeansImpl.java StudentBean.java StudentBeansImpl.java
1 package com.evergent.corejava.javabeans;
2
3 public class StudentBeansImpl {
4
5     public static void main(String[] args) {
6         StudentBean s=new StudentBean();
7         s.setSno(100);
8         s.setName("sandhya");
9         s.setAddress("hyderabad");
10
11         System.out.println(s);
12     }
13 }
14 }
15
```

Console ×

terminated> StudentBeansImpl [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot

student no: 100
student name: sandhya
student address: hyderabad

Collections Framework

ArrayList

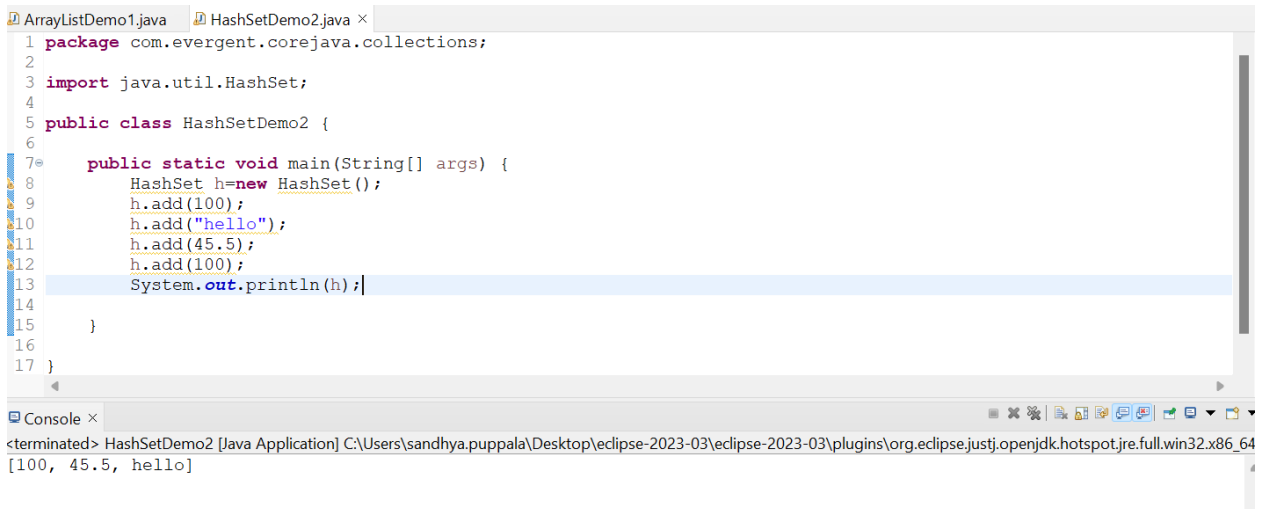


The screenshot shows the Eclipse IDE with a file named `ArrayListDemo1.java` open. The code defines a package `com.evergent.corejava.collections`, imports `java.util.ArrayList`, and creates a public class `ArrayListDemo1`. Inside the class, the `main` method initializes an `ArrayList` object, adds the elements `100`, `"hello"`, `45.5`, and `100` to it, and then prints the list. The console output shows the list as `[100, hello, 45.5, 100]`.

```
1 package com.evergent.corejava.collections;
2 import java.util.ArrayList;
3
4 public class ArrayListDemo1 {
5
6     public static void main(String[] args) {
7         ArrayList list=new ArrayList();
8         list.add(100);
9         list.add("hello");
10        list.add(45.5);
11        list.add(100);
12        System.out.println(list);
13    }
14 }
15
16
```

Console ×
<terminated> ArrayListDemo1 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_100, hello, 45.5, 100]

HashSet



The screenshot shows the Eclipse IDE with a file named `HashSetDemo2.java` open. The code defines a package `com.evergent.corejava.collections`, imports `java.util.HashSet`, and creates a public class `HashSetDemo2`. Inside the class, the `main` method initializes a `HashSet` object, adds the elements `100`, `"hello"`, `45.5`, and `100` to it, and then prints the set. The console output shows the set as `[100, 45.5, hello]`.

```
1 package com.evergent.corejava.collections;
2
3 import java.util.HashSet;
4
5 public class HashSetDemo2 {
6
7     public static void main(String[] args) {
8         HashSet h=new HashSet();
9         h.add(100);
10        h.add("hello");
11        h.add(45.5);
12        h.add(100);
13        System.out.println(h);
14    }
15 }
16
17
```

Console ×
<terminated> HashSetDemo2 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_100, 45.5, hello]

hasNext() program

```
ArrayListDemo3.java ×
1 package com.evergent.corejava.collections;
2
3 import java.util.ArrayList;
4 import java.util.Iterator;
5
6 public class ArrayListDemo3 {
7
8     public static void main(String[] args) {
9         ArrayList list=new ArrayList();
10        list.add(100);
11        list.add("hello");
12        list.add(45.5);
13        list.add(100);
14        System.out.println(list);
15        Iterator i=list.iterator();
16        while(i.hasNext())
17        {
18            System.out.println(i.next());
19        }
20    }
21 }
22 }
23 }
24 }
```

Console ×

```
<terminated> ArrayListDemo3 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
[100, hello, 45.5, 100]
100
hello
45.5
100
```

Program 4)

```
*HashSetDemo4.java × HashSetDemo2.java ArrayListDemo3.java
1 package com.evergent.corejava.collections;
2
3 import java.util.HashSet;
4 import java.util.Iterator;
5
6 public class HashSetDemo4 {
7
8     public static void main(String[] args) {
9         HashSet h=new HashSet();
10        h.add(100);
11        h.add("hello");
12        h.add(45.5);
13        h.add(100);
14        System.out.println(h);
15        Iterator i=h.iterator();
16        while(i.hasNext())
17        {
18            System.out.println(i.next());
19        }
20    }
21 }
22 }
```

Console ×

```
<terminated> HashSetDemo4 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
[100, 45.5, hello]
100
45.5
hello
```

Program 5)

TreeSet is for Ordering(Ascending order)

```
TreeSetDemo5.java ×
1 package com.evergent.corejava.collections;
2
3 import java.util.TreeSet;
4
5 public class TreeSetDemo5 {
6
7     public static void main(String[] args) {
8         TreeSet t=new TreeSet();
9         t.add(100);
10        t.add(5);
11        t.add(30);
12        t.add(500);
13        System.out.println(t);
14    }
15
16 }
17

Console ×
<terminated> TreeSetDemo5 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17
[5, 30, 100, 500]
```

Day 14

Wrapper Classes

Auto Boxing and Auto Unboxing

1)program

```
LibraryUser.java  LibraryInterfaceDemo.java  WrapperClassDemo1.java ×
1 package com.evergent.corejava.wrapperclasses;
2
3 public class WrapperClassDemo1 {
4
5     public static void main(String[] args) {
6         //Auto-Boxing
7         int a=10;
8         Integer il=new Integer(a);
9         System.out.println(il);
10
11        //Auto-Unboxing
12        int al=il.intValue();
13        System.out.println(al);
14
15    }
16
17 }
18

Console ×
<terminated> WrapperClassDemo1 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
10
10
```

2) Program

```

1 package com.evergent.corejava.wrapperclasses;
2 public class WrapperClassDemo2 {
3     public static void main(String[] args) {
4         float f1=4.5f;
5         Float f2=new Float(f1);
6         float f3=f2.floatValue();
7         double d1=799.89;
8         Double d2=new Double(d1);
9         double d3=d2.doubleValue();
10        byte b1=10;
11        Byte b2=new Byte(b1);
12        byte b3=b2.byteValue();
13        //float value
14        System.out.println("float value"+f1);
15        System.out.println("float object value "+f2);
16        System.out.println("convert float object value to primitive type:"+f3);
17
18        //double value
19        System.out.println("Double value"+d1);
20        System.out.println("Double object value "+d2);
21        System.out.println("convert double object value to primitive type:"+d3);
22
23        //Byte value
24        System.out.println("Byte value"+b1);
25        System.out.println("Byte object value "+b2);
26        System.out.println("convert Byte object value to primitive type:"+b3);
27    }
28 }

```

```

<terminated> WrapperClassDemo2 [Java Application] C:\Users\sandhya.puppala\Desktop\eclipse-2023-03\eclipse-2023-03\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
float value4.5
float object value 4.5
convert float object value to primitive type:4.5
Double value799.89
Double object value 799.89
convert double object value to primitive type:799.89
Byte value10
Byte object value 10
convert Byte object value to primitive type:10

```