

Assignment 3

1. Define a base class person and a derived class employee with single inheritance.

-Define SetData() member functions in each of the class with different signatures to set the data members and demonstrate overloading of member functions.

-Define GetData() member functions in each of the class with same signatures to display data and demonstrate overriding of member functions.

```
class Person{
    int id;
    String name;
    int age;
    String city;

    public void setData(int id,String name,int age,String city){
        this.id=id;
        this.name=name;
        this.age=age;
        this.city=city;
    }

    public void getData()
    { System.out.println("Person Id:"+id+"\nPerson Name:"+name+"\nPerson Age:"+age+"\nPerson
    City:"+city);}

}
```

```
class Employee extends Person{
    int sal;

    public void setData(int id,String name,int age,String city,int sal){
```

```
this.id=id;

this.name=name;

this.age=age;

this.city=city;

this.sal=sal;

}

public void getData()

{ System.out.println("Employee Id:"+id+"\nEmployee Name:"+name+"\nEmployee
Age:"+age+"\nEmployee City:"+city+"\nEmployee sal:"+sal);}


public static void main(String []args){

Person p=new Person();

System.out.println("Person class data");

System.out.println("#####");

p.setData(101,"Swati",23,"Delhi");

p.getData();

System.out.println();

System.out.println("Employee class Data");

System.out.println("#####");

Employee e=new Employee();

e.setData(100,"Prachi",23,"Mumbai",200000);

e.getData();

}

}
```

```
osboxes ~ > Java > OOP > assign3 > javac Employee.java
osboxes ~ > Java > OOP > assign3 > java Employee
Person class data
#####
Person Id:101
Person Name:Swati
Person Age:23
Person City:Delhi

Employee class Data
#####
Employee Id:100
Employee Name:Prachi
Employee Age:23
Employee City:Mumbai
Employee sal:200000
```

2. Modify program 1 to define a parametrized constructor and finalizer in each class.

Demonstrate calling the constructor of the base class from the constructor of the derived class.

-Create objects of person and employee classes to show the order of invocation of constructors.

```
class Person{
    int id;
    String name;
    int age;
    String city;

    public Person(int id,String name,int age,String city){
        this.id=id;
        this.name=name;
        this.age=age;
        this.city=city;
    }

    public void display()
    { System.out.println("Person Id:"+id+"\nPerson Name:"+name+"\nPerson Age:"+age+"\nPerson
    City:"+city);}
```

```
}
```

```
class Employee1 extends Person{
```

```
int sal;
```

```
public Employee1(int id,String name,int age,String city,int sal){
```

```
super(id,name,age,city);
```

```
this.sal=sal;
```

```
}
```

```
public void display()
```

```
{ System.out.println("Employee Id:"+id+"\nEmployee Name:"+name+"\nEmployee  
Age:"+age+"\nEmployee City:"+city+"\nEmployee Salary:"+sal);}
```

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

```
Person p=new Person(100,"Swati",23,"Delhi");
```

```
System.out.println("Base class data");
```

```
System.out.println("#####");
```

```
p.display();
```

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

```
System.out.println("Derived class Data");
```

```
System.out.println("#####");
```

```
Employee1 e=new Employee1(102,"Prachi",23,"Delhi",20000);
```

```
e.display();
```

```
}
```

```
}
```

```
osboxes ~ > Java > OOP > assign3 > 1 > java Employee1
Base class data
#####
Person Id:100
Person Name:Swati
Person Age:23
Person City:Delhi

Derived class Data
#####
Employee Id:102
Employee Name:Prachi
Employee Age:23
Employee City:Delhi
Employee Salary:20000
```

3. Modify program 2 to define another class manager that derives from employee, to create a chain of multilevel

hierarchy.(manager inherits from employee & employee inherits from person)

-Create objects of person, employee, and manager classes to show the order of invocation of constructors.

```
class Person{
    int id;
    String name;
    int age;
    String city;

    public Person(int id,String name,int age,String city){
        this.id=id;
        this.name=name;
        this.age=age;
        this.city=city;
    }

    public void display()
    { System.out.println("Person Id:"+id+"\nPerson Name:"+name+"\nPerson Age:"+age+"\nPerson
    City:"+city);}
```

```
}
```

```
class Employee extends Person{
```

```
int sal;
```

```
public Employee(int id,String name,int age,String city,int sal){
```

```
super(id,name,age,city);
```

```
this.sal=sal;
```

```
}
```

```
public void display()
```

```
{ System.out.println("Employee Id:"+id+"\nEmployee Name:"+name+"\nEmployee  
Age:"+age+"\nEmployee City:"+city+"\nEmployee Salary:"+sal);}
```

```
}
```

```
class Manager extends Employee{
```

```
int bonus=1000;
```

```
// bonus=sal+bonus;
```

```
public Manager(int id,String name,int age,String city,int sal,int bonus){
```

```
super(id,name,age,city,sal);
```

```
this.bonus=bonus;
```

```
}
```

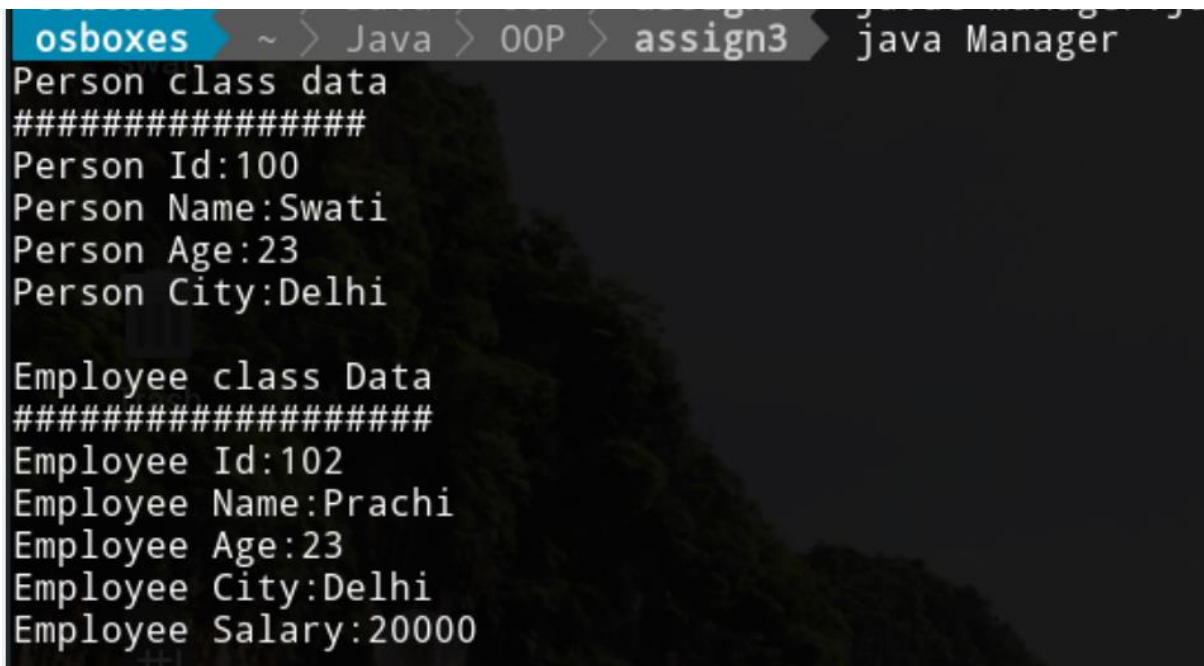
```
public void display()
```

```
{ System.out.println("Manager Id:"+id+"\nManager Name:"+name+"\nManager  
Age:"+age+"\nManager City:"+city+"\nManager Salary:"+sal+"\nManager Bonus:"+bonus);}
```

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

```
Person p=new Person(100,"Swati",23,"Delhi");
```

```
System.out.println("Person class data");
System.out.println("#####");
p.display();
System.out.println();
System.out.println("Employee class Data");
System.out.println("#####");
Employee e=new Employee(102,"Prachi",23,"Delhi",20000);
e.display();
System.out.println();
System.out.println("Manager Class Data");
System.out.println("#####");
Manager m=new Manager(1004,"Mahesh",45,"Gurugram",20000,21000);
m.display();
}
}
```



```
osboxes ~ > Java > OOP > assign3 > java Manager
Person class data
#####
Person Id:100
Person Name:Swati
Person Age:23
Person City:Delhi

Employee class Data
#####
Employee Id:102
Employee Name:Prachi
Employee Age:23
Employee City:Delhi
Employee Salary:20000
```

```
Manager Class Data
#####
Manager Id:1004
Manager Name:Mahesh
Manager Age:45
Manager City:Gurugram
Manager Salary:20000
Manager Bonus:21000
```

4. Modify program 2 to define another class student that derives from person, to create a hierarchical

inheritance.(employee and student inherit from person)

-Create objects of person, employee, and student classes to show the order of invocation of constructors.

```
class Person{
    int id;
    String name;
    int age;
    String city;

    public Person(int id,String name,int age,String city){
        this.id=id;
        this.name=name;
        this.age=age;
        this.city=city;
    }

    public void display()
    { System.out.println("Person Id:"+id+"\nPerson Name:"+name+"\nPerson Age:"+age+"\nPerson
    City:"+city);}

}
```



```
class Employee extends Person{
    int sal;

    public Employee(int id,String name,int age,String city,int sal){
        super(id,name,age,city);
        this.sal=sal;
    }

    public void display()
    { System.out.println("Employee Id:"+id+"\nEmployee Name:"+name+"\nEmployee
    Age:"+age+"\nEmployee City:"+city+"\nEmployee Salary:"+sal);}
}

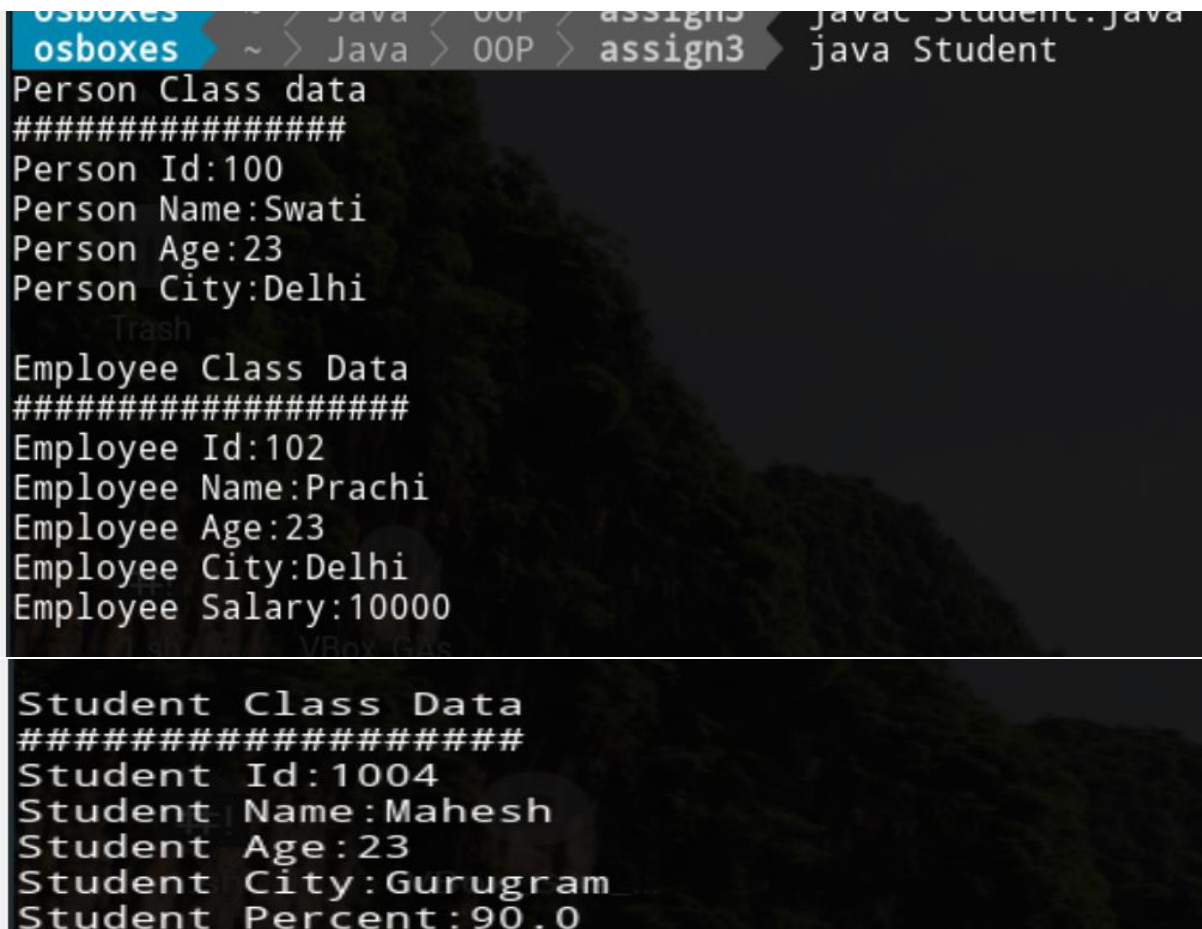
class Student extends Person{
    float percent;

    public Student(int id,String name,int age,String city,float percent){
        super(id,name,age,city);
        this.percent=percent;
    }

    public void display()
    { System.out.println("Student Id:"+id+"\nStudent Name:"+name+"\nStudent Age:"+age+"\nStudent
    City:"+city+"\nStudent Percent:"+percent);}

    public static void main(String []args){
        Person p=new Person(100,"Swati",23,"Delhi");
        System.out.println("Person Class data");
        System.out.println("#####");
        p.display();
        System.out.println();
    }
}
```

```
System.out.println("Employee Class Data");  
System.out.println("#####");  
Employee e=new Employee(102,"Prachi",23,"Delhi",10000);  
e.display();  
System.out.println();  
System.out.println("Student Class Data");  
System.out.println("#####");  
Student s=new Student(1004,"Mahesh",23,"Gurugram",90);  
s.display();  
}  
}
```



```
osboxes ~ > Java > OOP > assign3 > java Student.java  
Person Class data  
#####  
Person Id:100  
Person Name:Swati  
Person Age:23  
Person City:Delhi  
Employee Class Data  
#####  
Employee Id:102  
Employee Name:Prachi  
Employee Age:23  
Employee City:Delhi  
Employee Salary:10000  
Student Class Data  
#####  
Student Id:1004  
Student Name:Mahesh  
Student Age:23  
Student City:Gurugram  
Student Percent:90.0
```

5. Write a java file Person.java containing definition of independent class Person in package com.person,
and another java file Address.java containing definition of independent class Address in com.address

package.

-Write a container class employee that has a person and an address object as contained data members (besides other data members) by importing the above packages.

-Demonstrate the passing of arguments in the constructors of person and address classes by the constructor of the employee class.

//file Person.java

```
package com.person;
```

```
public class Person{
```

```
    int id;
```

```
    int age;
```

```
    String name;
```

```
    String city;
```

```
    public Person(int id,int age,String name,String city){
```

```
        this.id=id;
```

```
        this.age=age;
```

```
        this.name=name;
```

```
        this.city=city;
```

```
    }
```

```
    public void display(){
```

```
        System.out.println("Person Detail");
```

```
        System.out.println("#####");
```

```
        System.out.println("Person Id:"+id+"\nPerson age:"+age+"\nPerson Name:"+name+"\nPerson City:"+city);
```

```
    }
```

```
}
```

//File Address.java

```
package com.address;
```

```
public class Address{  
    int hno;  
    String street;  
    String city;  
    int pin;  
  
    public Address(int hno,String street,String city,int pin){  
        this.hno=hno;  
        this.street=street;  
        this.city=city;  
        this.pin=pin;  
    }  
    public void display(){  
        System.out.println("Address Detail");  
        System.out.println("#####");  
        System.out.println("H.No:"+hno+"\nStreet:"+street+"\nCity:"+city+"\nPincode:"+pin);  
    }  
}
```

//file Employee.java

```
package mypack;  
import com.address.Address;  
import com.person.Person;  
public class Employee{  
    int id;  
    int age;  
    String name;  
    Person p;  
    Address add;  
    Employee(int id,int age,String name,Person p,Address add)  
    {  
        this.id=id;
```

```
this.age=age;
this.name=name;
this.p=p;
this.add=add;
}
void display()
{
System.out.println("Employee Data:");
System.out.println("#####");
System.out.println("Employee Id:"+id+"\nEmployee Age:"+age+"\nEmployee Name:"+name);
System.out.println();
p.display();
System.out.println();
add.display();
}
public static void main(String[]args){
Person p=new Person(100,23,"Prachi","Delhi");
Address a=new Address(789,"SainVihar","Delhi",100200);

Employee e=new Employee(1002,23,"Komal",p,a);
e.display();
}
}
```

```

osboxes ... > 00P > assign3 > Q5 2 javac -d . Employee.java
osboxes ... > 00P > assign3 > Q5 java mypack.Employee

Employee Data:
#####
Employee Id:1002
Employee Age:23
Employee Name:Komal

Person Detail
#####
Person Id:100
Person age:23
Person Name:Prachi
Person City:Delhi

Address Detail
#####
H.No:789
Street:SainVihar
City:Delhi
Pincode:100200

```

6. Rework program 5 to use both inheritance and containership. The employee class inherits from person class and acts as a container class for address class object.

//File Address.java

```

package com.address;

public class Address{

    int hno;

    String street;

    String city;

    int pin;

    public Address(int hno,String street,String city,int pin){

        this.hno=hno;

        this.street=street;

        this.city=city;

        this.pin=pin;

    }

    public void display(){

```

```
System.out.println("Address Detail");  
System.out.println("#####");  
System.out.println("H.No:"+hno+"\nStreet:"+street+"\nCity:"+city+"\nPincode:"+pin);  
}  
}
```

//File Employee.java

```
package mypack;  
import com.address.Address;  
class Person{  
    int id;  
    int age;  
    String name;  
    Address add;  
  
    public Person(int id,int age,String name,Address add){  
        this.id=id;  
        this.age=age;  
        this.name=name;  
        this.add=add;  
    }  
    public void display(){  
        System.out.println("Person Detail");  
        System.out.println("#####");  
        System.out.println("Person Id:"+id+"\nPerson age:"+age+"\nPerson Name:"+name);  
        add.display();  
    }  
    public static void main(String []args){  
        Address a=new Address(789,"SainVihar","Delhi",100200);
```

```
Person p=new Person(100,23,"Prachi",a);  
}  
}
```

```
public class Employee extends Person{  
    int sal;
```

```
    Employee(int id,int age,String name,Address add,int sal)
```

```
{  
    super(id,age,name,add);  
    this.sal=sal;
```

```
}  
    public void display()
```

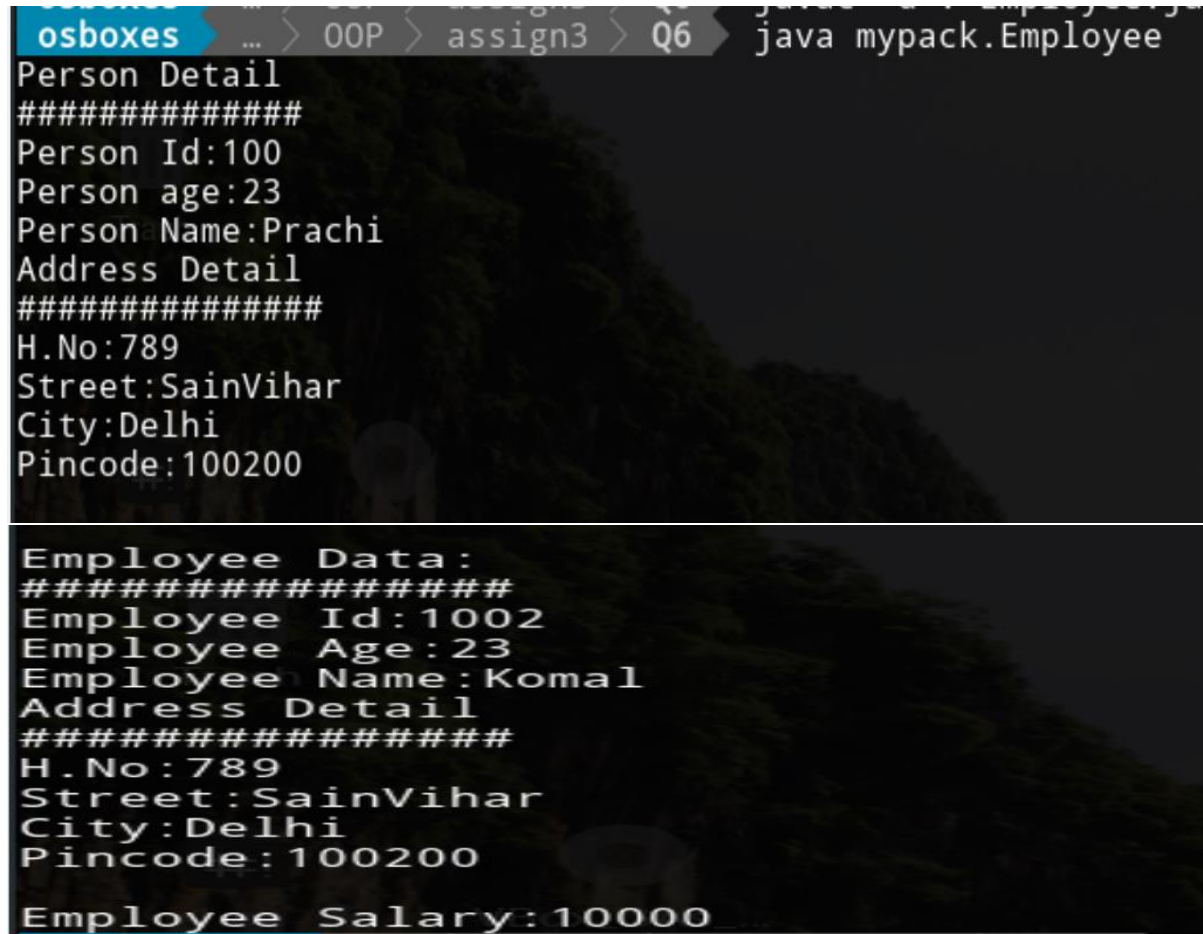
```
{  
    System.out.println("Employee Data:");  
    System.out.println("#####");  
    System.out.println("Employee Id:"+id+"\nEmployee Age:"+age+"\nEmployee Name:"+name);  
    System.out.println();  
    add.display();  
    System.out.println("Employee Salary:"+sal);
```

```
}
```

```
    public static void main(String[]args){  
        //Address a=new Address(789,"SainVihar","Delhi",100200);  
        //Person p=new Person(100,23,"Prachi",a);  
        Address a=new Address(789,"SainVihar","Delhi",100200);  
        Person p=new Person(100,23,"Prachi",a);  
        p.display();  
        System.out.println();  
        Employee e=new Employee(1002,23,"Komal",a,10000);
```



```
e.display();  
}  
}
```



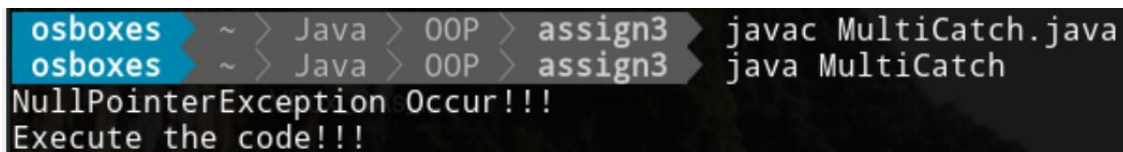
```
osboxes > ... > 00P > assign3 > Q6 > java mypack.Employee  
Person Detail  
#####  
Person Id:100  
Person age:23  
Person Name:Prachi  
Address Detail  
#####  
H.No:789  
Street:SainVihar  
City:Delhi  
Pincode:100200  
  
Employee Data:  
#####  
Employee Id:1002  
Employee Age:23  
Employee Name:Komal  
Address Detail  
#####  
H.No:789  
Street:SainVihar  
City:Delhi  
Pincode:100200  
  
Employee Salary:10000
```

7. Write a program to demonstrate the use of try, catch, finally throw and throws keywords and demonstrate the following points in the program.

a) Multiple catch blocks.

```
public class MultiCatch {  
  
    public static void main(String[] args) {  
  
        try{  
            String s=null;  
            System.out.println(s.length());  
        }  
  
        catch(ArithmeticException e)
```

```
{
    System.out.println("Arithmetic Exception occurs!!!\n");
}
catch(ArrayIndexOutOfBoundsException e)
{
    System.out.println("ArrayIndexOutOfBoundsException occurs!!!\n");
}
catch(NullPointerException e)
{
    System.out.print("NullPointerException Occur!!!\n");
}
catch(Exception e)
{
    System.out.println("Super class exception Occur!!!");
}
System.out.println("Execute the code!!!");
}
}
```



A terminal window showing the compilation and execution of a Java program. The prompt is 'osboxes ~ >'. The user enters 'Java > OOP > assign3' followed by 'javac MultiCatch.java'. The prompt changes to 'osboxes ~ > Java > OOP > assign3' and the user enters 'java MultiCatch'. The output shows 'NullPointerException Occur!!!' and 'Execute the code!!!'.

b) try-catch-finally combination.

```
class Combination {
    public static void main (String args[]) {
        //try block
        try
        {

            int data = 125 / 0;

            System.out.println ("Result:" + data);
        }
    }
}
```

```
//catch block
catch (Exception e) {

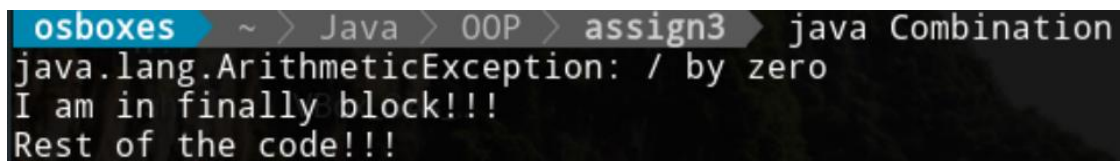
    System.out.println (e);
}

//finally block
finally {

    System.out.println ("I am in finally block!!!");

}

System.out.println ("Rest of the code...");
}
}
```

A terminal window with a dark background and light-colored text. The prompt is 'osboxes ~ > Java > OOP > assign3'. The command 'java Combination' has been executed. The output shows an exception: 'java.lang.ArithmeticException: / by zero', followed by 'I am in finally block!!!' and 'Rest of the code!!!'.

```
osboxes ~ > Java > OOP > assign3 java Combination
java.lang.ArithmeticException: / by zero
I am in finally block!!!
Rest of the code!!!
```

c) try-finally combination.

```
class Finally{
public static void main(String [] args){
try{
int num=70/0;
System.out.println(num);
}
finally{
System.out.println("Hello I am in Finally block!!!");
}
System.out.println("Hello I am out of block");
}
}
```

```
osboxes ~ > Java > OOP > assign3 javac Finally.java
osboxes ~ > Java > OOP > assign3 java Finally
Hello I am in Finally block!!!
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Finally.main(Finally.java:4)
```

d) Exception propagation among many methods.

```
class Propagation{
```

```
    public void division(int n1, int n2) {
```

```
        System.out.println(n1/n2);
```

```
    }
```

```
    public void method1(int n1, int n2) {
```

```
        //not caught here and hence propagate to method2.
```

```
        division(n1,n2);
```

```
    }
```

```
    public void method2(int n1, int n2){
```

```
        try{
```

```
            method1(n1,n2);
```

```
        }catch(ArithmeticException e){
```

```
            System.out.println(e);
```

```
            System.out.println("Exception Handled");
```

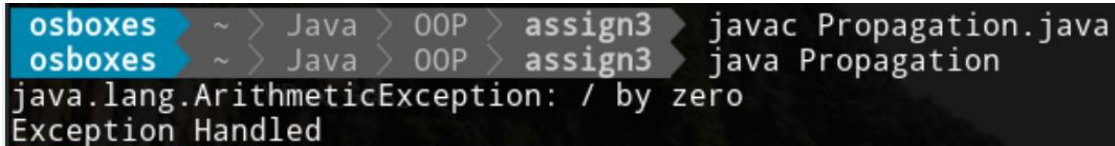
```
        }
```

```
    }
```

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

```
Propagation p = new Propagation();

//method call
p.method2(20, 0);
}
}
```



```
osboxes ~ > Java > OOP > assign3 > javac Propagation.java
osboxes ~ > Java > OOP > assign3 > java Propagation
java.lang.ArithmeticException: / by zero
Exception Handled
```

e) Use of getMessage(), printStackTrace() function of Throwable class.

```
import java.io.*;

public class Throw {

    public static void method() {

        try

        {

            FileReader file = new FileReader("C:\\Users\\Prachi\\Desktop\\abc.txt");

            BufferedReader fileInput = new BufferedReader(file);

            throw new FileNotFoundException();

        } catch (FileNotFoundException e)

        { System.out.println(e);

            System.out.println("Output for getMessage() method :");

            System.out.println("#####");

            e.getMessage();

            System.out.println("Output for printStackTrace() method:");

        }

        System.out.println("Exception Handle Succesfully");

    }

    public static void main(String args[]){

        method();

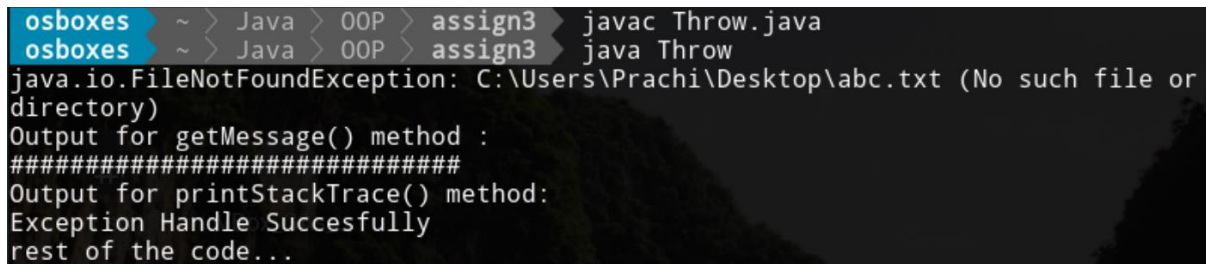
        System.out.println("rest of the code...");

    }

}
```

```
}
```

```
}
```



```
osboxes ~ > Java > OOP > assign3 javac Throw.java
osboxes ~ > Java > OOP > assign3 java Throw
java.io.FileNotFoundException: C:\Users\Prachi\Desktop\abc.txt (No such file or
directory)
Output for getMessage() method :
#####
Output for printStackTrace() method:
Exception Handle Succesfully
rest of the code...
```

```
import java.io.*;
```

```
public class Throw {
```

```
public static void method() {
```

```
    try
```

```
    {
```

```
        FileReader file = new FileReader("C:\\Users\\Prachi\\Desktop\\abc.txt");
```

```
        BufferedReader fileInput = new BufferedReader(file);
```

```
        throw new FileNotFoundException();
```

```
    } catch (FileNotFoundException e)
```

```
    {
```

```
        System.out.println("Output for printStackTrace() method:");
```

```
        System.out.println("#####");
```

```
        e.printStackTrace();
```

```
    }
```

```
    System.out.println("Exception Handle Succesfully");
```

```
    }
```

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

```
        method();
```

```
        System.out.println("rest of the code...");
```

```
    }
```

```
}
```

```
osboxes ~ > Java > OOP > assign3 javac Throw.java
osboxes ~ > Java > OOP > assign3 java Throw
Output for printStackTrace() method:
#####
java.io.FileNotFoundException: C:\Users\Prachi\Desktop\abc.txt (No such file or
directory)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:212)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:154)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:109)
    at java.base/java.io.FileReader.<init>(FileReader.java:60)
    at Throw.method(Throw.java:7)
    at Throw.main(Throw.java:21)
Exception Handle Succesfully
rest of the code...
```

f) Nested try blocks

```
public class NestedTry{

    public static void main(String args[]){

        try{

            //inner try block1

            try{

                int a =3/0;

            }

            //catch block of inner try block 1

            catch(ArithmeticException e)

            {

                System.out.println(e);

            }

            //inner try block 2

            try{

                int a[]=new int[5];

                a[5]=4;

            }

            //catch block of inner try block 2

            catch(ArrayIndexOutOfBoundsException e)

            {

                System.out.println(e);

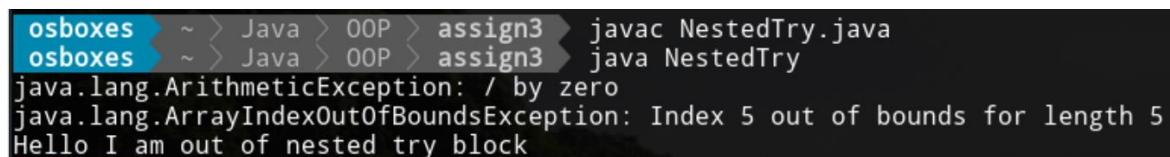
            }

        }

    }

}
```

```
    }  
}  
  
//catch block of outer try block  
catch(Exception e)  
{  
    System.out.println("handled the exception (outer catch)");  
}  
  
System.out.println("Hello I am out of nested try block");  
}  
}
```



```
osboxes ~ > Java > OOP > assign3 > javac NestedTry.java  
osboxes ~ > Java > OOP > assign3 > java NestedTry  
java.lang.ArithmeticException: / by zero  
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5  
Hello I am out of nested try block
```

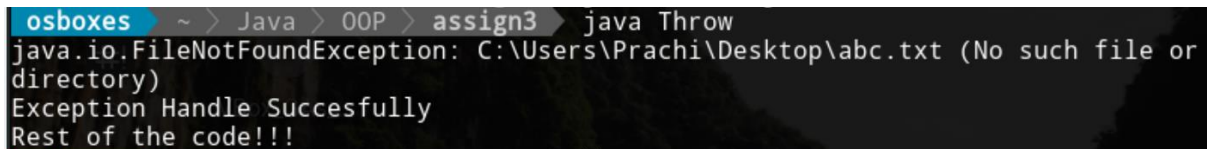
8. Write a program to throw a checked exception explicitly using 'throw' keyword and

a) Handle the exception in same method.

```
import java.io.*;  
  
public class Throw {  
    public static void method() {  
        try  
        {  
  
            FileReader file = new FileReader("C:\\Users\\Prachi\\Desktop\\abc.txt");  
            BufferedReader fileInput = new BufferedReader(file);  
            throw new FileNotFoundException();  
        } catch (FileNotFoundException e)  
        {  
            System.out.println(e);  
        }  
  
        System.out.println("Exception Handle Succesfully");  
    }  
}
```



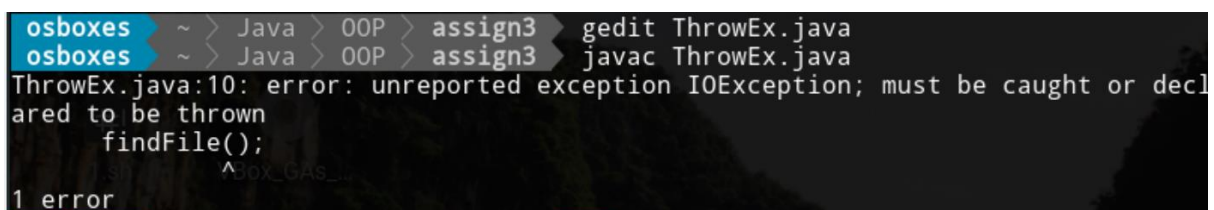
```
}  
  
public static void main(String args[]){  
    method();  
  
    System.out.println("Rest of the code!!!");  
}  
}
```



```
osboxes ~ > Java > OOP > assign3 > java Throw  
java.io.FileNotFoundException: C:\Users\Prachi\Desktop\abc.txt (No such file or  
directory)  
Exception Handle Succesfully  
Rest of the code!!!
```

b) use throws clause and handle the exception in some other method (calling method)

```
import java.io.*;  
  
class ThrowEx {  
    public static void findFile() throws IOException {  
        System.out.println("Rest of code in try block");  
        throw new IOException("File not found");  
    }  
  
    public static void main(String[] args) {  
        try {  
            findFile();  
            System.out.println("Rest of code in try block");  
        } catch (IOException e) {  
            System.out.println(e);  
        }  
    }  
}
```



```
osboxes ~ > Java > OOP > assign3 > gedit ThrowEx.java  
osboxes ~ > Java > OOP > assign3 > javac ThrowEx.java  
ThrowEx.java:10: error: unreported exception IOException; must be caught or decl  
ared to be thrown  
    findFile();  
    ^  
1 error
```

c) Don't either handle or use the throws clause. Observe the result.

```
import java.io.*;

class ThrowEx {

    public static void findFile() throws IOException {

        System.out.println("hello");

    }

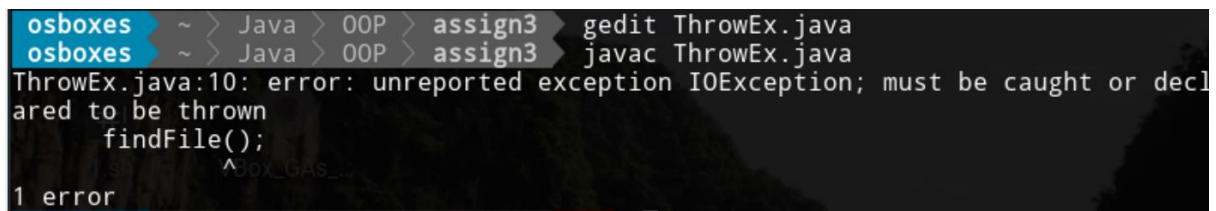
    public static void main(String[] args) {

        findFile();

        System.out.println("Rest of code in try block");

    }

}
```



```
osboxes ~ > Java > OOP > assign3 gedit ThrowEx.java
osboxes ~ > Java > OOP > assign3 javac ThrowEx.java
ThrowEx.java:10: error: unreported exception IOException; must be caught or declared to be thrown
    findFile();
    ^
1 error
```

9. Repeat program 8 with unchecked Exception and demonstrate the difference in both program.

```
public class UncheckedException {

    public static void main(String[] args)

    {

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

        // Try block for exceptions

        try {

            System.out.println(a[5]);

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Array out of index");

        }

    }

}
```

```
    }  
}  
}
```

```
osboxes ~ > Java > OOP > assign3 javac UncheckedException.java  
osboxes ~ > Java > OOP > assign3 java UncheckedException  
Array out of index
```

b) Handle unchecked exception in other method or calling method.

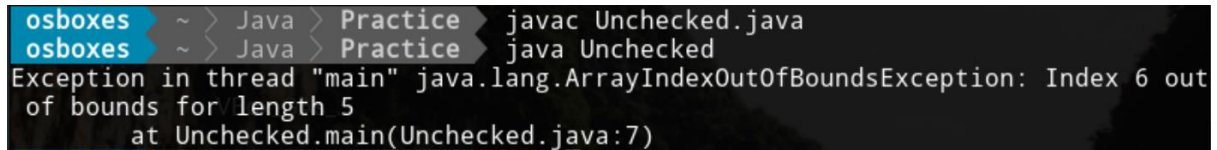
```
public class Unchecked {  
    static void ExceptionHandle(){  
        int a[] = {1,2,3,4,5};  
  
        // Try block for exceptions  
        try {  
            System.out.println(a[5]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println(e);  
            System.out.println("Array out of index");  
        }  
    }  
}  
  
public static void main(String[] args)  
{ ExceptionHandle();  
  
}  
}
```

```
osboxes ~ > Java > Practice javac Unchecked.java  
osboxes ~ > Java > Practice java Unchecked  
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5  
Array out of index
```

c) Don't either handle or use the throws clause. Observe the result.

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

```
//ArrayIndexOutOfBoundsException  
int array[] = {1, 2, 3, 4, 5};  
System.out.println(array[6]);  
}  
}
```



```
osboxes ~ > Java > Practice > javac Unchecked.java  
osboxes ~ > Java > Practice > java Unchecked  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 6 out  
of bounds for length 5  
    at Unchecked.main(Unchecked.java:7)
```

Difference of checked and Unchecked Exception Program

I observe that checked Exception is caught by compiler at the time of program compile not the run time of program

Unchecked Exception caught by Interpreter at the time of program execution.

10. Create a user defined exception to check whether your employee exist in your data structure (use any data structure to store the employees -like array, ArrayList etc) and throw exception if name is not in the employees list. Use the catch and finally block to make an appropriate solution.

```
import java.io.*;  
import java.util.*;  
public class Employee{  
  
    public static void main(String []args)throws IOException{  
        int i=0;  
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  
        System.out.print("Enter The Employee Name:");  
        String name=br.readLine();  
        ArrayList<String> a = new ArrayList<>();  
  
        a.add("Prachi");  
        a.add("Rakesh");  
        a.add("Sapna");  
        a.add("Priya");
```

```
a.add("megha");  
if (!a.contains(name)) {  
  
    try {  
        throw new NameNotFoundException(name);  
    } catch (NameNotFoundException e) {  
        System.out.println(e);  
        System.out.println("Sorry!!! Name is not found!!!");  
    }  
    finally{  
        br.close();  
        System.out.println("All resource are closed!!!");  
    }  
}  
else  
{  
    System.out.println("Employee name is Found!!!");  
}  
}  
}
```

//file NameNotFoundException.java

```
public class NameNotFoundException extends Exception{  
    public NameNotFoundException(String errorMsg)  
    {  
        super(errorMsg);  
    }  
}
```

```
osboxes ~ > Java > Practice javac Employee.java
osboxes ~ > Java > Practice java Employee
Enter The Employee Name:Prachi
Employee name is Found!!!
osboxes ~ > Java > Practice java Employee
Enter The Employee Name:Vicky
NameNotFoundException: Vicky
Sorry!!! Name is not found!!!
All resource are closed!!!
```

11. Write a program to demonstrate the use of equals method of Object class and compare its functionality with (==) operator.

```
public class Equal {
    public static void main(String[] args)
    {
        String s1 = "HELLO";
        String s2 = "HELLO";
        String s3 = new String("HELLO");
        System.out.println("== operator used to compare address of objects");
        System.out.println("#####");
        System.out.println(s1 == s2);
        System.out.println(s1 == s3);
        System.out.println("equals() method used to compare contents of objects");
        System.out.println("#####");
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
    }
}
```

```
osboxes ~ > Java > OOP > assign3 > javac Equal.java
osboxes ~ > Java > OOP > assign3 > java Equal
== operator used to compare address of objects
#####
true
false
equals() method used to compare contents of objects
#####
true
true
```

Difference between == and equals() method.

- 1 .equals() method is a method and == is operator.
- 2 .equals() method used for content comparison and == operators used for reference comparison (address comparison)

12. Modify program 1 with Integer class Object. Use the override equals method of Integer with two different object wrapping same primitive int value (like -10).demonstrate the difference in the output from program 11.

```
import java.lang.*;

class Int{

public static void main(String[]args){

Integer a=new Integer(-10);

Integer b=new Integer(-10);

System.out.println("Compare content of object using equals() method");

System.out.println("#####");

System.out.println(a.equals(b));

System.out.println(a.equals(b));

System.out.println(b.equals(a));

Integer c=new Integer(10);

System.out.println(c.equals(a));

System.out.println(c.equals(b));

System.out.println("Compare references of object using == operator");

System.out.println("#####");

System.out.println(a==b);

System.out.println(a==c);

}
```

}

```
osboxes ~ > Java > OOP > assign3 java Int
Compare content of object using equals() method
#####
true
true
true
false
false
Compare references of object using == operator
#####
false
false
```

All wrapper class Objects are immutable.

- this equals() method override in the String class objects

so String class equal() method performed the content comparison

- The equals() method is a method of Integer class under java.lang package.

This method compares the value of the parameter to the value of the current Integer object.

It returns Boolean which corresponds to the equality of this Integer and method argument object.

It also overrides the equals() method of Object class.

13. Demonstrate the use of ceil(), floor(), round(), random(), abs(), max(), min() methods of Math class.

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

        int a = 20;
        int b = -54;
        double d1 = 84.6;
        double d2 = 6.45;
        //math.abs() method;
        System.out.println();
        System.out.println("abs() method");
```



```
System.out.println("#####");
System.out.println("Absolute value of a:" + Math.abs(a));
System.out.println("Absolute value of b:" + Math.abs(b));
System.out.println("Absolute value of d1:" + Math.abs(d1));
System.out.println("Absolute value of d2:" + Math.abs(d2));

//math.round() method
System.out.println();
System.out.println("Round() method");
System.out.println("#####");
System.out.println("Round off for d1:"+Math.round(d1));
System.out.println("Round off for d2:"+Math.round(d2));

//math.ceil() and floor()method
System.out.println();

System.out.println("ceil() and floor()method");
System.out.println("#####");
System.out.println("Ceiling of "+d1+"="+Math.ceil(d1));
System.out.println("Floor of "+d1+"="+ Math.floor(d1));
System.out.println("Ceiling of"+d2+"="+Math.ceil(d2));
System.out.println("Floor of"+d2+"="+Math.floor(d2));

//math.min() and math.max() method
System.out.println();
System.out.println("Minimum and Maximum method");
System.out.println("#####");
System.out.println("Minimum out of"+a+" and "+b+"="+Math.min(a,b));
System.out.println("Maximum out of"+b+" and "+b+"="+Math.max(a,b));
System.out.println("Minimum out of"+d1+" and "+d2+"="+Math.min(d1,d2));
System.out.println("Maximum out of"+d1+" and "+d2+"="+Math.max(d1,d2));
```

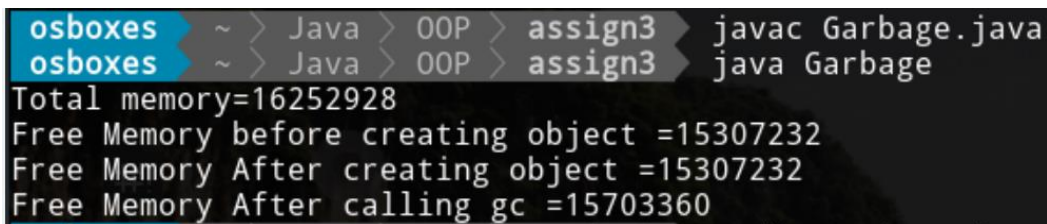
```
//math.random() method  
System.out.println();  
System.out.println("Random() method");  
System.out.println("#####");  
System.out.println("Random doubles:"+Math.random());  
System.out.println("Random doubles:"+Math.random());  
int c = (int)(Math.random()*(100));  
System.out.println("Random() method on integer:"+c);  
}
```

```
}  
osboxes ~ > Java > OOP > assign3 > java MathMethod  
abs() method  
#####  
Absolute value of a:20  
Absolute value of b:54  
Absolute value of d1:84.6  
Absolute value of d2:6.45  
  
Round() method  
#####  
Round off for d1:85  
Round off for d2:6  
  
ceil() and floor()method  
#####  
Ceiling of 84.6=85.0  
Floor of 84.6=84.0  
Ceiling of 6.45=7.0  
Floor of 6.45=6.0  
  
Minimum and Maximum method  
#####  
Minimum out of 20 and -54=-54  
Maximum out of -54 and 20=20  
Minimum out of 84.6 and 6.45=6.45  
Maximum out of 84.6 and 6.45=84.6  
  
Random() method  
#####  
Random doubles:0.10717898219602118  
Random doubles:0.4445373351235816  
Random() method on integer:76
```

14. Write a program to invoke garbage collector and show the details of free memory before and after the garbage collection.

```
import java.util.*;  
  
class Garbage  
{  
  
public static void main(String[] args)  
{
```

```
Garbage g=new Garbage();  
Runtime r=Runtime.getRuntime();  
System.out.println("Total memory="+r.totalMemory());  
System.out.println("Free Memory before creating object "+r.freeMemory());  
for(int i=0;i<10;i++)  
{  
    Date d=new Date();  
    d=null;  
}  
System.out.println("Free Memory After creating object "+r.freeMemory());  
r.gc();  
System.out.println("Free Memory After calling gc "+r.freeMemory());  
}  
}
```



```
osboxes ~ > Java > OOP > assign3 javac Garbage.java  
osboxes ~ > Java > OOP > assign3 java Garbage  
Total memory=16252928  
Free Memory before creating object =15307232  
Free Memory After creating object =15307232  
Free Memory After calling gc =15703360
```

15. Write a program to print all System properties using system class.

```
import java.util.Properties;  
  
public class PrintSystem{  
  
    public static void main(String[] args){  
  
        // List all System properties  
  
        Properties pros = System.getProperties();  
        pros.list(System.out);  
    }  
}
```

```
osboxes ~ > Java > OOP > assign3 javac PrintSystem.java
osboxes ~ > Java > OOP > assign3 java PrintSystem
-- listing properties --
java.specification.version=14
sun.management.compiler=HotSpot 64-Bit Tiered Compilers
sun.jnu.encoding=UTF-8
java.runtime.version=14.0.2+12-Ubuntu-120.04
java.class.path=.
user.name=osboxes
java.vm.vendor=Private Build
path.separator=:
sun.arch.data.model=64
os.version=5.4.0-70-generic
java.runtime.name=OpenJDK Runtime Environment
file.encoding=UTF-8
java.vendor.url=Unknown
java.vm.name=OpenJDK 64-Bit Server VM
java.vm.specification.version=14
os.name=Linux
sun.java.launcher=SUN_STANDARD
user.country=US
sun.boot.library.path=/usr/lib/jvm/java-14-openjdk-amd64/lib
sun.java.command=PrintSystem
java.vendor.url.bug=Unknown
java.io.tmpdir=/tmp
jdk.debug=release
sun.cpu.endian=little
java.version=14.0.2
user.home=/home/osboxes
user.dir=/home/osboxes/Java/OOP/assign3
os.arch=amd64
user.language=en
java.specification.vendor=Oracle Corporation
java.vm.specification.name=Java Virtual Machine Specification
java.version.date=2020-07-14
java.home=/usr/lib/jvm/java-14-openjdk-amd64
file.separator=/
java.vm.compressedOopsMode=32-bit
line.separator=
Trash
java.library.path=/usr/java/packages/lib:/usr/lib/x86_64...
java.vm.info=mixed mode, sharing
java.vm.specification.vendor=Oracle Corporation
java.specification.name=Java Platform API Specification
java.vendor=Private Build
java.vm.version=14.0.2+12-Ubuntu-120.04
sun.io.unicode.encoding=UnicodeLittle
java.class.version=58.0
```

16. Define a method setMyProperty (String, String) to set your own system property and use the same system property in another method

```
import java.util.Properties;

class SetProperty{

public static void main(String [] args){

    //to set system property

    System.setProperty("java.home", "\\home\\Prachi");

    System.setProperty("user.name", "Prachi");

    System.setProperty("user.dir", "\\Java\\OOP");


    //to get property of system

    System.out.println("Get java home");

    System.out.println("#####");

    //System.out.println();

    System.out.println(System.getProperty("java.home"));

    System.out.println();

    System.out.println("Get user name");

    System.out.println("#####");

    System.out.println(System.getProperty("user.name"));

    System.out.println();

    System.out.println("Get user Directory");

    System.out.println("#####");

    System.out.println(System.getProperty("user.dir"));

}

}
```

```
osboxes ~ > Java > OOP > assign3 > javac SetProperty.java
osboxes ~ > Java > OOP > assign3 > java SetProperty
Get java home
#####
\home\Prachi

Get user name
#####
Prachi

Get user Directory
#####
\Java\OOP
```