**Notes: First you prepare how to write a code for syntax and write a code**

**Java: (day one)**

Steps to install java

Steps to install eclipse

Steps to create workspace

Ans: https://www.tutorialspoint.com/eclipse/eclipse\_workspaces.htm

Steps to create project

File -> Project

We call Project is program

Ans: <https://www.tutorialspoint.com/eclipse/eclipse_create_java_project.htm>

create .java file/class

Project -> new class and give extension .java

When you create a class it will create class name with keyword class and name of class

Class will contain method and properties

Class Employee

{ //begin of class

} //end of class

how to create packages and what is best way to give name

From solution explorer, select project, right click and select package

<https://www.tutorialspoint.com/eclipse/eclipse_create_java_package.htm>

Ex: companyname.projectname.foldername (this is common naming standard)

ex:chase.gems.DB

what is main method will do?

Main method is starting point of program

creating property/data members : we create properties at class level

Properties will store the value

To create variable we specify

Datatype Variablename;

int salary

String firstname

what is data type and different data types`

It will represents what type of data we are storing

Int

Double

Float

String

bool

What is variable?

It will store the value in memory

To create variable we specify

Datatype Variablename;

creating method with void : we write methods in class

Inside method We write code/steps for manual test case steps

Method syntax :

**Returntype methodname parenthesis**

Ex: Void MethodName()

{

}

Method contains two parts:

Ex: Void MethodName(int param1) //method declaration/signature

{ //method body

}

//creating method with void

Ex: Void MethodName2() //method declaration/signature

{ //method body

}

If we put “void” method don’t return anything from method

creating variable, we can create variables inside method

Variable scope is within method only

Ex: int MethodName()

{

Int var1;

}

creating method with return data type, we can return **int/string/double/float/date** etc

When we are returning from method we have to use **return** keyword inside method

After return key you can return hard coded value:

Return 2

Or

return propertyname

Or

return variablename

**Note:** value that we specify after **return** keyword should be of data type that is specified in method signature

//method that will return hard coded value

Ex: int MethodName()

{

Return 2 //hardcoded value

}

//method that will return property value

Ex: int MethodName()

{

Return empId //property value

}

//method that will return property value

Ex: int MethodName()

{

Int var1=2;

Return var1; // return variable

}

//creating method with return data type and parameter

int MethodName(int id) {

Int id;

return id;

}

creating static property: All instances shared the value http://crunchify.com/java-static-methods-variables-static-block-and-class-with-example/

Static int empid;

creating static method

http://crunchify.com/java-static-methods-variables-static-block-and-class-with-example/

//static method

Static void GetEmpid()

{

}

creating object

calling method with void

calling method with no return and parameter

public class Bicycle {

    private int cadence;

    private int gear;

    private int speed;

    public Bicycle(int startCadence, int startSpeed, int startGear) {

        gear = startGear;

        cadence = startCadence;

        speed = startSpeed;

    }

    public int getCadence() {

        return cadence;

    }

    public void setCadence(int newValue) {

        cadence = newValue;

    }

    public int getGear() {

        return gear;

    }

    public void setGear(int newValue) {

        gear = newValue;

    }

    public int getSpeed() {

        return speed;

    }

    public void applyBrake(int decrement) {

        speed -= decrement;

    }

    public void speedUp(int increment) {

        speed += increment;

    }

  public static void main(String[] args) {

        Bicycle st = new Bicycle(20,10,4);

     st.setGear(2);

     int a = st.getGear();

    System.out.println("Result: " + a);

    }

}

calling static method

using static property: it will maintain

**Ans:** public static int empid;

create classes under multiple packages

calling classes under different packages

write code to handle exceptions with try/catch/finally

public void divide(int numberToDivide, int numberToDivideBy)

throws BadNumberException{

if(numberToDivideBy == 0){

throw new BadNumberException("Cannot divide by 0");

}

return numberToDivide / numberToDivideBy;

}

Catching Exceptions:

public void callDivide(){

try {

int result = divide(2,1);

System.out.println(result);

} catch (BadNumberException e) {

//do something clever with the exception

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

}

System.out.println("Division attempt done");

}

what is final keyword, create final class, final method, final property

write code for interface and create class to implement that interface

write code for creating abstract class

|  |
| --- |
| import main.java.music.ElectricBassGuitar; |

|  |  |
| --- | --- |
|  | import main.java.music.ElectricGuitar; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | public class Execution { |

|  |  |
| --- | --- |
| 5 |  |

|  |  |
| --- | --- |
|  | public static void main(String[] args) { |

|  |  |
| --- | --- |
|  | ElectricGuitar guitar = new ElectricGuitar(); |

|  |  |
| --- | --- |
|  | ElectricBassGuitar bassGuitar = new ElectricBassGuitar(); |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | guitar.play(); |

|  |  |
| --- | --- |
|  | bassGuitar.play(); |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | guitar = new ElectricGuitar(7); |

|  |  |
| --- | --- |
|  | bassGuitar = new ElectricBassGuitar(5); |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | guitar.play(); |

|  |  |
| --- | --- |
|  | bassGuitar.play(); |

|  |  |
| --- | --- |
|  | } |

|  |
| --- |
| } |

implement method overloading

class overLoading

{

int add(int n1, int n2)

{

return n1+n2;

}

int add(int n1, int n2, int n3)

{

return n1+n2+n3;

}

int add(int n1, int n2, int n3, int n4)

{

return n1+n2+n3+n4;

}

int add(int n1, int n2, int n3, int n4, int n5)

{

return n1+n2+n3+n4+n5; }

public static void main(String args[])

{

Sum obj = new Sum();

System.out.println("Sum of two numbers: "+obj.add(20, 21));

System.out.println("Sum of three numbers: "+obj.add(20, 21, 22));

System.out.println("Sum of four numbers: "+obj.add(20, 21, 22, 23));

System.out.println("Sum of five numbers: "+obj.add(20, 21, 22, 23, 24));

}

}

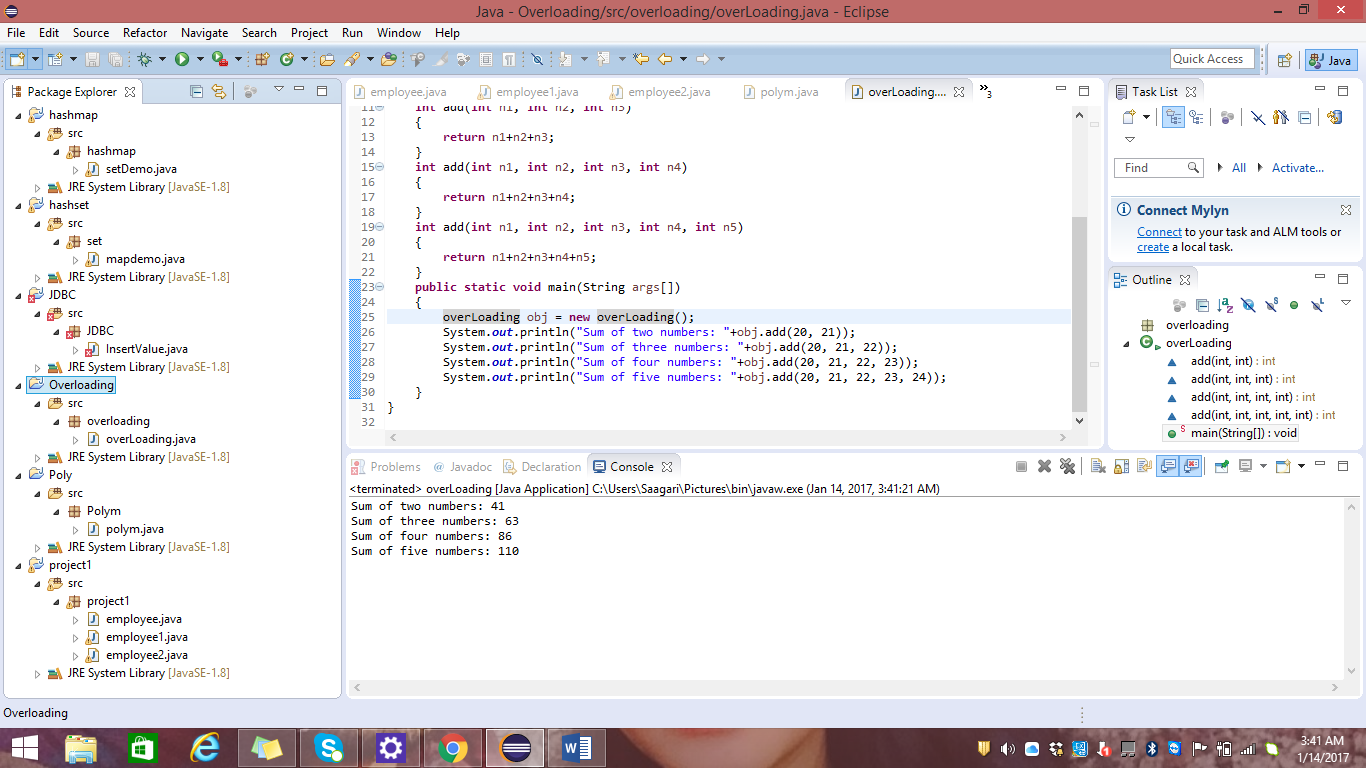
Output :

Sum of two numbers: 41

Sum of three numbers: 63

Sum of four numbers: 86

Sum of five numbers: 110



implement method overriding

package beginnersbook.com;

class CarClass

{

public int speedLimit()

{

return 100;

}

}

class Ford extends CarClass

{

public int speedLimit()

{

return 150;

}

public static void main(String args[])

{

CarClass obj = new Ford();

int num= obj.speedLimit();

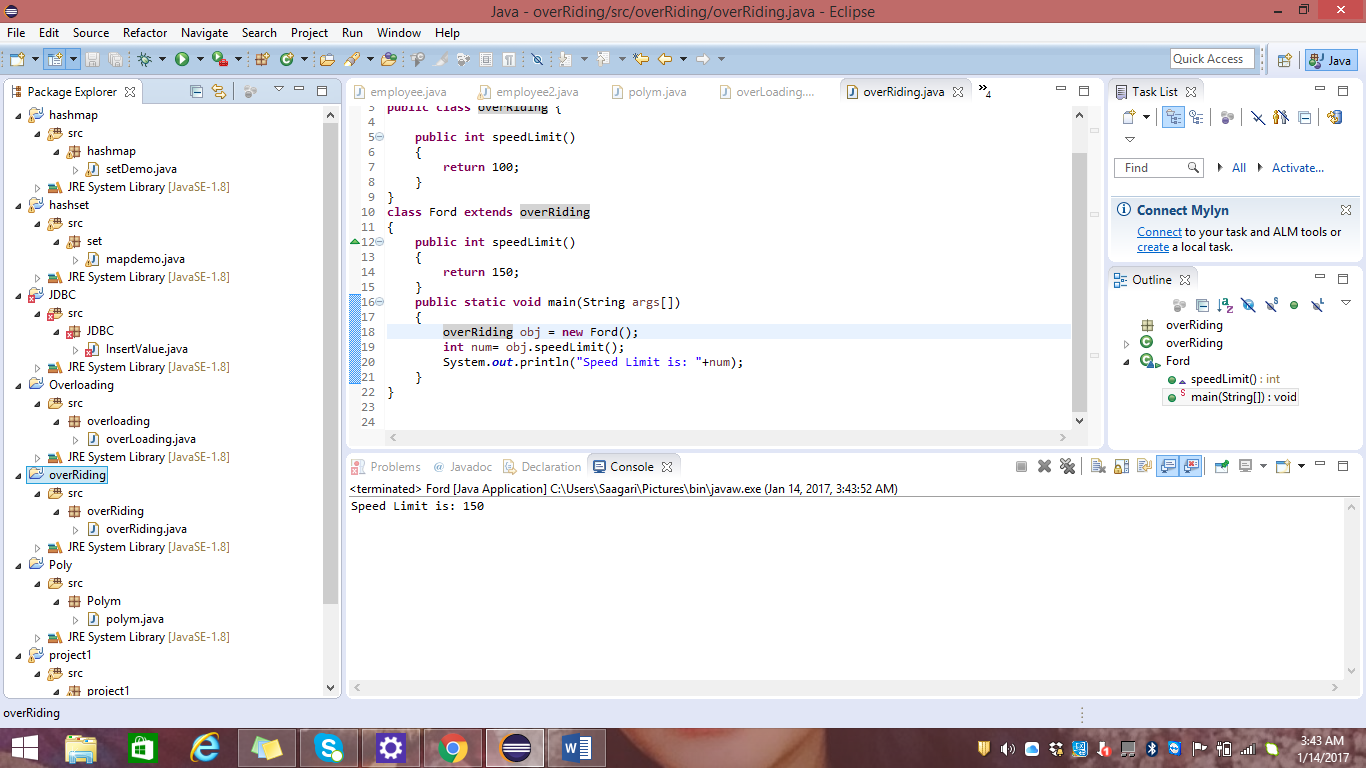
System.out.println("Speed Limit is: "+num);

}

}

OUTPUT:

Speed Limit is: 150



implementing polymorphism

Code:

class Polym //create main class

{

//main method

public static void main(String[] args)

{

//create object of class Student

Children chstud = new Student();

chstud.play();//call method

}

}

interface Children//create interface

{

public void play();//declare method

}

class Student implements Children//implements interface

{

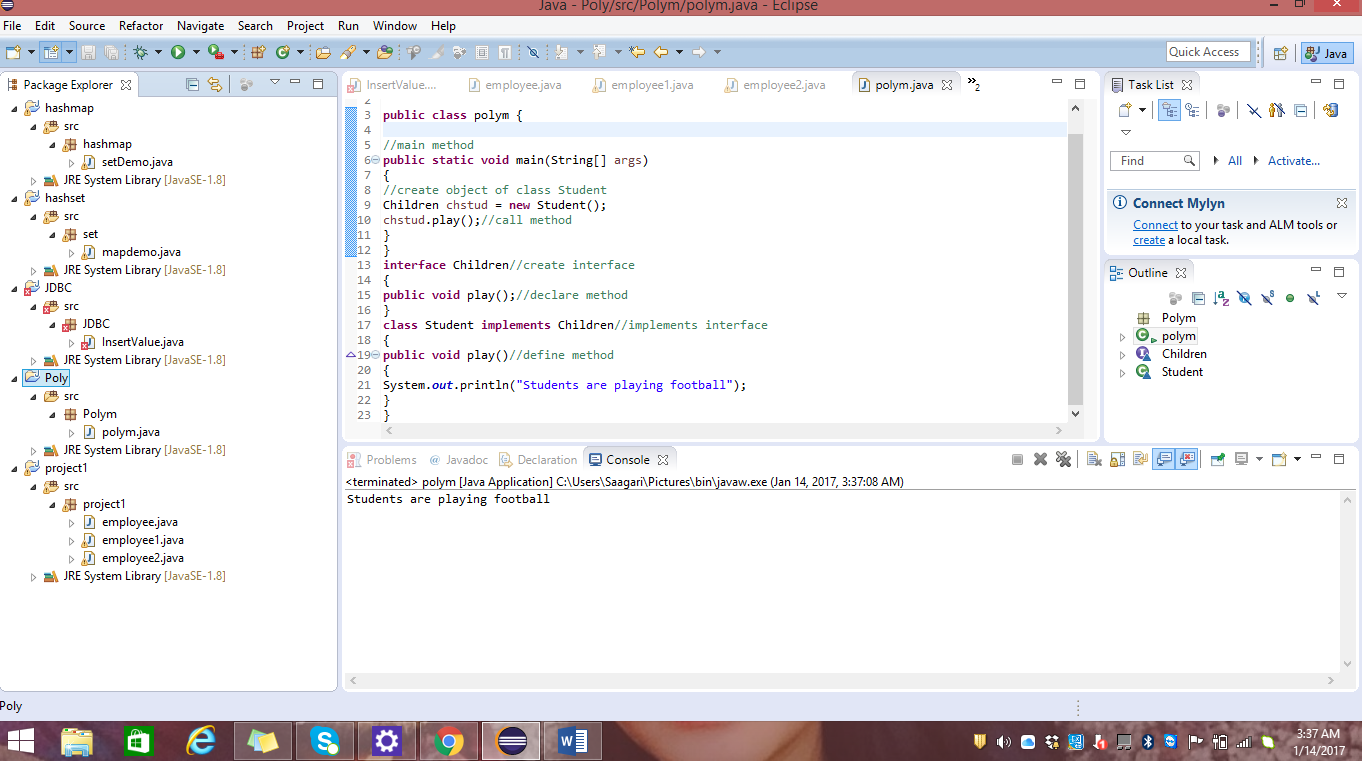
public void play()//define method

{

System.out.println("Students are playing football");

}

}



implementing interface

How to do inheritance in java (using extend keyword)

public class Bicycle {  
   
 // **the Bicycle class has three *fields***  
 public int cadence;  
 public int gear;  
 public int speed;  
   
 // **the Bicycle class has one *constructor***  
 public Bicycle(int startCadence, int startSpeed, int startGear) {  
 gear = startGear;  
 cadence = startCadence;  
 speed = startSpeed;  
 }  
   
 // **the Bicycle class has four *methods***  
 public void setCadence(int newValue) {  
 cadence = newValue;  
 }  
   
 public void setGear(int newValue) {  
 gear = newValue;  
 }  
   
 public void applyBrake(int decrement) {  
 speed -= decrement;  
 }  
   
 public void speedUp(int increment) {  
 speed += increment;  
 }  
   
}

A class declaration for a MountainBike class that is a subclass of Bicycle might look like this:

public class MountainBike extends Bicycle {  
   
 // **the MountainBike subclass adds one *field***  
 public int seatHeight;  
  
 // **the MountainBike subclass has one *constructor***  
 public MountainBike(int startHeight,  
 int startCadence,  
 int startSpeed,  
 int startGear) {  
 super(startCadence, startSpeed, startGear);  
 seatHeight = startHeight;  
 }   
   
 // **the MountainBike subclass adds one *method***  
 public void setHeight(int newValue) {  
 seatHeight = newValue;  
 }   
}

==================================+=

write code to add items to integer, string **array**

public static void main(String[] args) {

int[] series = new int[0];

int x = 5;

series = addInt(series, x);

//print out the array with commas as delimiters

System.out.print("New series: ");

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

if (i == series.length - 1){

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

}

else{

System.out.print(series[i] + ", ");

}

}

}

write code to retrieve items from integer, string **array**

Write code to add items to ArrayList collection and retrieve items to ArrayList collection.

import java.util.ArrayList;

import java.util.Collections;

import java.util.Enumeration;

import java.util.Iterator;

import java.util.List;

public class ArrayListDemo

{

    public static void main(String args[])

    {

        List list = new ArrayList();

        //Adding values to the ArrayList

        list.add("test1");

        list.add("test2");

        list.add("test3");

        System.out.println("Retrieving values from ArrayList using Iterator");

        retrieveValuesFromListMethod1(list);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        System.out.println("Retrieving values from ArrayList using get method");

        retrieveValuesFromListMethod2(list);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        System.out.println("Retrieving values from ArrayList using Enumeration");

        retrieveValuesFromListMethod3(list);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

    }

    /\*This method retrieves values from ArrayList using Iterator

     \*/

    public static void retrieveValuesFromListMethod1(List list)

    {

        Iterator itr = list.iterator();

        while(itr.hasNext())

        {

            System.out.println(itr.next());

        }

    }

/\*This method retrieves values from ArrayList using get method

     \*/

    public static void retrieveValuesFromListMethod2(List list)

    {

        //Retrieving values from list

        int size = list.size();

        for(int i=0;i<size;i++)

        {

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

        }

    }

    /\*This method retrieves values from ArrayList using Enumeration

     \*/

    public static void retrieveValuesFromListMethod3(List list)

    {

        Enumeration e = Collections.enumeration(list);

        while(e.hasMoreElements())

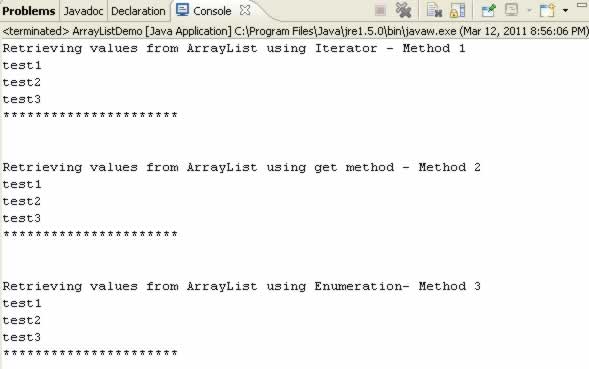
        {

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

        }

    }

}



write code to add items HashMap and retrieve items HashMap

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

public class MapDemo

{

    public static void main(String args[])

    {

        Map map = new HashMap();

        //Adding values to the HashMap

        map.put("test key 1", "test value 1");

        map.put("test key 2", "test value 2");

        map.put("test key 3", "test value 3");

        System.out.println("Retrieving values from HashMap");

        retrieveValuesFromListMethod(map);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

    }

    /\*This method retrieves values from Map

     \*/

    public static void retrieveValuesFromListMethod(Map map)

    {

        Set keys = map.keySet();

        Iterator itr = keys.iterator();

        String key;

        String value;

        while(itr.hasNext())

        {

            key = (String)itr.next();

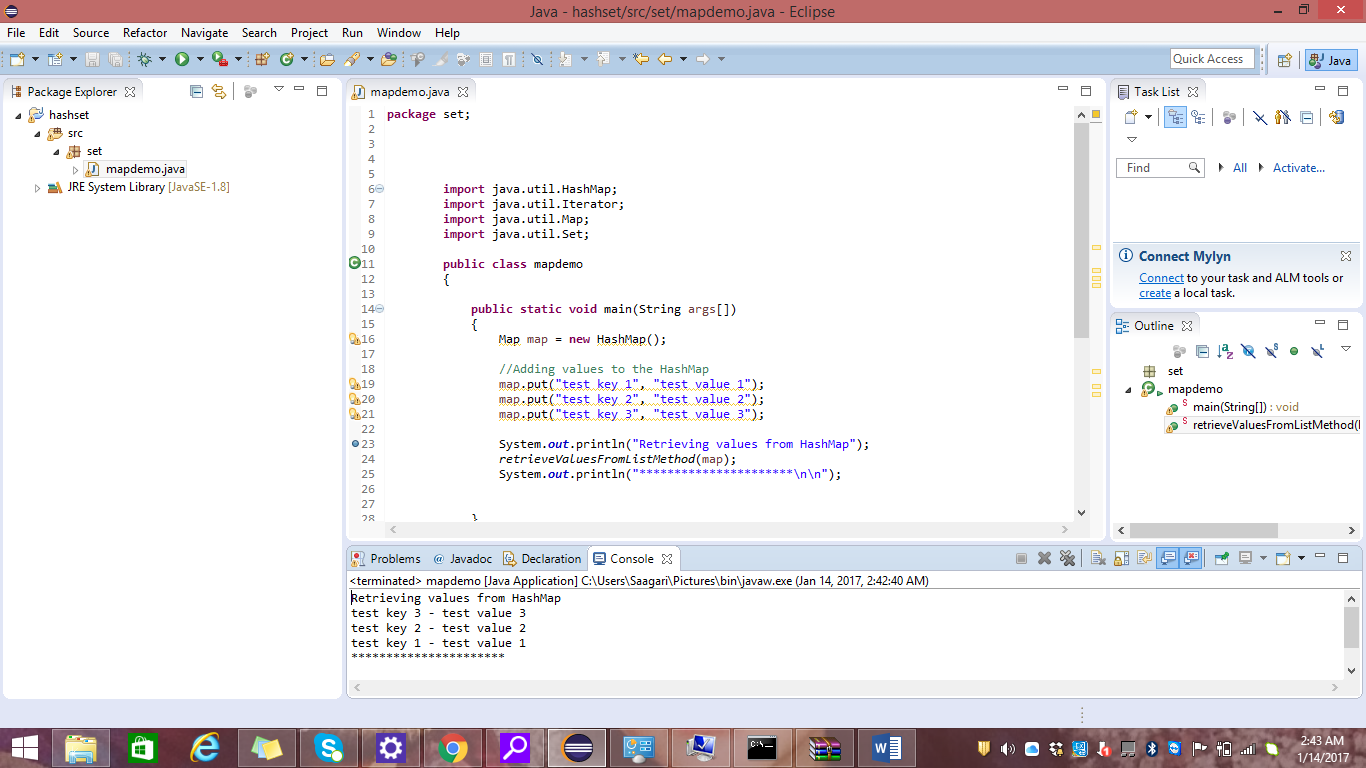
            value = (String)map.get(key);

            System.out.println(key + " - "+ value);

        }

    }

}



Write code to add items to hashset and retrieve items to hashset

import java.util.Collections;

import java.util.Enumeration;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

public class SetDemo

{

    public static void main(String args[])

    {

        Set set = new HashSet();

        //Adding values to the HashSet

        set.add("test1");

        set.add("test2");

        set.add("test3");

        System.out.println("Retrieving values from HashSet using Iterator");

        retrieveValuesFromListMethod1(set);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        System.out.println("Retrieving values from HashSet using Enumeration");

        retrieveValuesFromListMethod2(set);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

    }

    /\*This method retrieves values from HashSet using Iterator

     \*/

    public static void retrieveValuesFromListMethod1(Set set)

    {

        Iterator itr = set.iterator();

        while(itr.hasNext())

        {

            System.out.println(itr.next());

        }

    }

    /\*This method retrieves values from HashSet using Enumeration

     \*/

    public static void retrieveValuesFromListMethod2(Set set)

    {

        Enumeration e = Collections.enumeration(set);

        while(e.hasMoreElements())

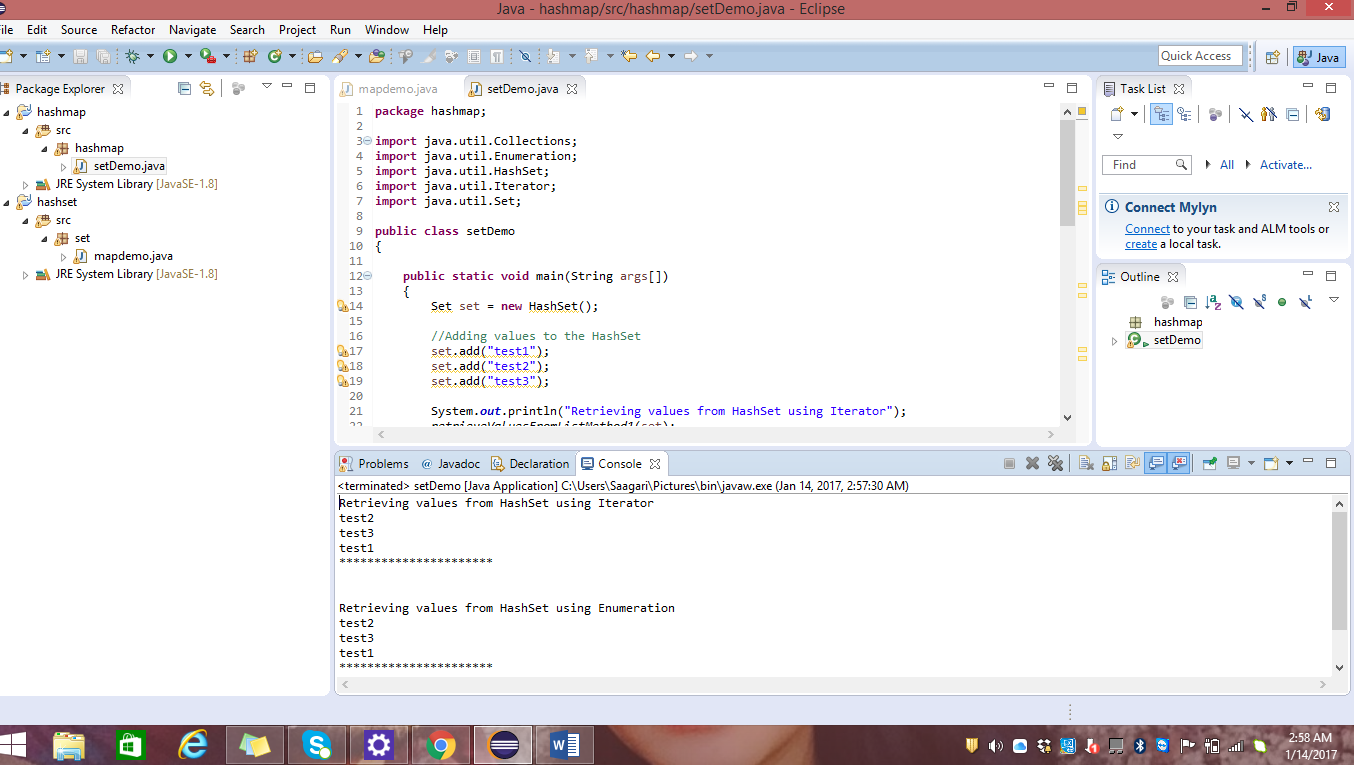
        {

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

        }

    }

}



write code to connect to JDBC to get rows from employee table

import java.sql.\*;

public class InsertValue {

        public static void main(String[] args) {

                 String url = " jdbc:odbc:dumy ";

     try {

      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            Connection con = DriverManager.getConnection("jdbc:odbc:dumy"; "sa", "");

            Statement stmt = con.createStatement();

            ResultSet reset = stmt.execute("select ename,job from emp");

            //Print the data to the console

            while(reset.next()){

                      System.out.println(reset.getString(1));

                      System.out.println(reset.getString(2));

            }

    catch( Exception e ) {

      e.printStackTrace();

    }

  }

}

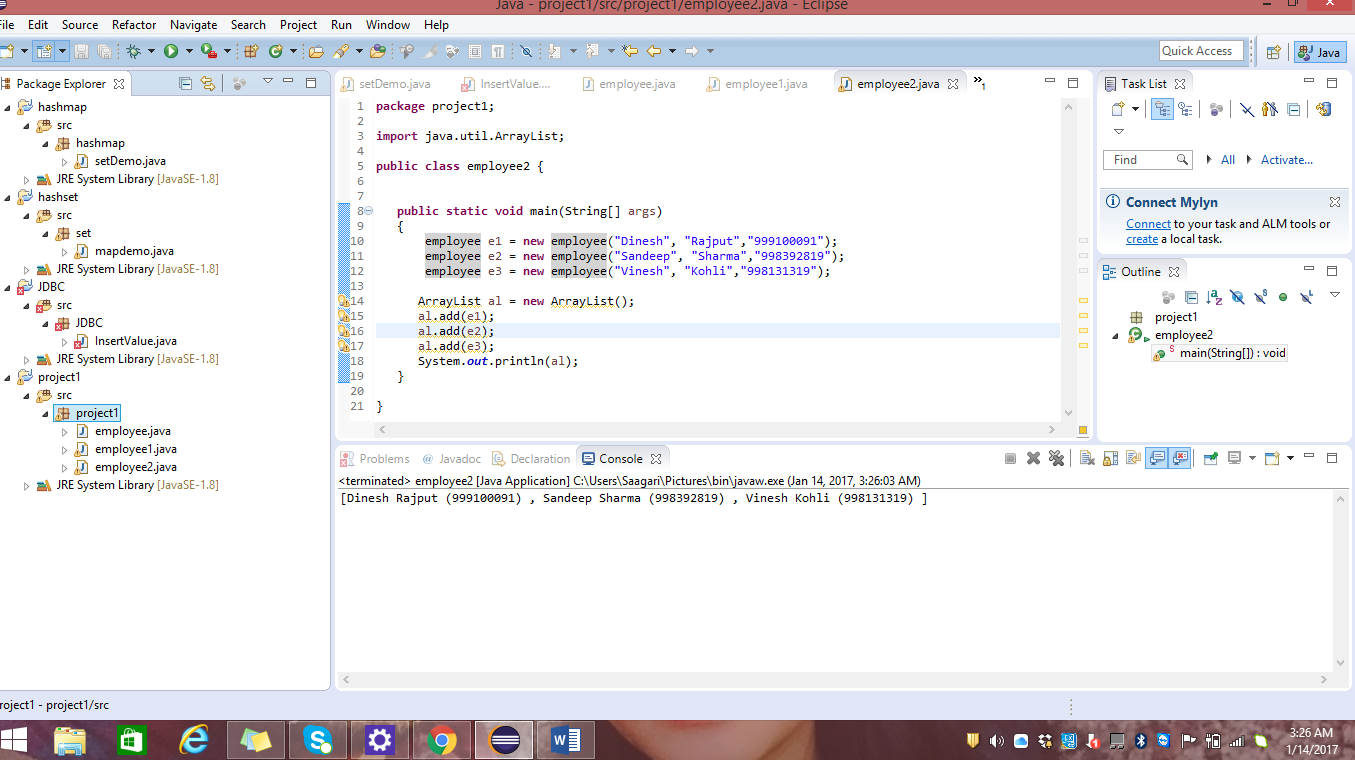
Write method to return list of rows code to loop throughs

create Employee class

1. **class** Employee
2. {
3. String firstName;
4. String lastName;
5. String phone;
6. **public** Employee(String firstName,String lastName,String phone)
7. {
8. **this**.firstName = firstName ;
9. **this**.lastName = lastName;
10. **this**.phone = phone;
11. }
13. **public** String toString()
14. {
15. **return** firstName +" "+lastName +" ("+phone +") ";
16. }
17. }

Add employee class to list collection

1. **public** **class** EmployeeListDemo
2. {
4. **public** **static** **void** main(String[] args)
5. {
6. Employee e1 = **new** Employee("Dinesh", "Rajput","999100091");
7. Employee e2 = **new** Employee("Sandeep", "Sharma","998392819");
8. Employee e3 = **new** Employee("Vinesh", "Kohli","998131319");
10. ArrayList al = **new** ArrayList();
11. al.add(e1);
12. al.add(e2);
13. al.add(e3);
14. System.**out**.println(al);
15. }
17. }



create method that return list of employee collection

Difference between string, string buffer, string builder with example

**String**  
  
Stringis *immutable*  ( once created can not be changed )object  . The object created as a String is stored in the  **Constant String Pool**  .   
Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.  
String  once assigned can not be changed.  
  
String  demo = " hello " ;  
// The above object is stored in constant string pool and its value can not be modified.  
  
  
demo="Bye" ;     //new "Bye" string is created in constant pool and referenced by the demo variable              
 // "hello" string still exists in string constant pool and its value is not overrided but we lost reference to the  "hello"string    
  
**StringBuffer**  
  
StringBufferis mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap . StringBuffer  has the same methods as the StringBuilder , but **each method in StringBuffer is synchronized**that is **StringBuffer is thread safe** .   
  
Due to this it does not allow  two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .  
  
But being thread safe has disadvantages too as the performance of the StringBuffer hits due to thread safe property . Thus  StringBuilder is faster than the StringBuffer when calling the same methods of each class.  
  
StringBuffer value can be changed , it means it can be assigned to the new value . Nowadays its a most common interview question ,the differences between the above classes .  
String Buffer can be converted to the string by using   
toString() method.  
  
StringBuffer demo1 = new StringBuffer("Hello") ;  
// The above object stored in heap and its value can be changed .

demo1=new StringBuffer("Bye");  
// Above statement is right as it modifies the value which is allowed in the StringBuffer  
  
**StringBuilder**  
  
StringBuilder  is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is that**StringBuilder is also not thread safe.**  
StringBuilder is fast as it is not thread safe .    
  
  
StringBuilder demo2= new StringBuilder("Hello");  
// The above object too is stored in the heap and its value can be modified  
demo2=new StringBuilder("Bye");   
// Above statement is right as it modifies the value which is allowed in the StringBuilder  
  
  
----------------------------------------------------------------------------------  
                                  ***String***                ***StringBuffer***        ***StringBuilder***  
----------------------------------------------------------------------------------                   
**Storage Area** | Constant String Pool           Heap                       Heap   
**Modifiable**     |  No (immutable)            Yes( mutable )          Yes( mutable )  
**Thread Safe**   |           Yes                                  Yes                              No  
**Performance** |         Fast                                Very slow                    Fast  
-----------------------------------------------------------------------------------

write a code to save data into excel file and read from excel file (POI and jexcel API)

how to update the data into XML file and read data from XML file