Foreign key: A foreign key is a column (or columns) that references a column (most often the primary key) of another table. The purpose of the foreign key is to ensure referential integrity of the data

Lets take person table with coloumns as P\_id,P\_firstname,P\_lastname.Here P\_id is the primary key in this table

|  |  |  |
| --- | --- | --- |
| P\_id | P\_firstname | P\_lastname |
| 1 | Sriram | Ajay |
| 2 | rash | bandari |
| 3 | naksha | vis |
| 4 | dolly | mami |

Lets take orders table .Here P\_Id is the foreign key in this table.

|  |  |  |
| --- | --- | --- |
| Order\_id | Order\_No. | P\_id |
| 1 | 4432 | 2 |
| 2 | 5321 | 1 |

SQL:

CREATE TABLE ORDERS (

ORDER\_ID INTEGER PRIMARY KEY,

ORDER\_NO INTEGER,

P\_ID INTEGER FOREIGN KEY REFERENCES PERSON(P\_ID)

);

GROUP BY:

select order\_no,count(order\_id) from orders group by oder\_no;

Having:

Store table

Store\_Name Sales Txn\_Date

Los Angeles 1500 Jan-05-1999

San Diego 250 Jan-07-1999

Los Angeles 300 Jan-08-1999

Boston 700 Jan-08-1999

select store\_name,sum(sales) from store group by store\_name having sum(sales)>1500;

Inner join:

select person.p\_id,person.p\_firstname,person.p\_lastname,orders.order\_id from person inner join orders on person.p\_id=orders.p\_id order by person.p\_firstname;

Left join:

select person.p\_id,person.p\_firstname,person.p\_lastname,orders.order\_id from person left join orders on person.p\_id=orders.p\_id order by person.p\_firstname;

max

select max(sales) from store;

select sum(sales) from store;

select avg(sales) from sales;

primary key:

CREATE TABLE PERSON (

P\_ID INTEGER PRIMARY KEY,

P\_FIRSTNAME VARCHAR(244),

P\_LASTNAME VARCHAR(244),

);

Where:

select \* from person where p\_firstname like ’s%’;

2nd highest:

Select max(sales) from store where sales not in (select max(sales) from store);

Table:consiste of rows and columns

Database:storage of data in the form of tables.

Steps to install eclipse:

Install jdk->download eclipse ->create shortcut on desktop

Static:

package student;

1. class Student{
2. int rollno;
3. String name;
4. static String college ="ITS";
5. Student8(int r,String n){
6. rollno = r;
7. name = n;
8. }
9. void display (){System.out.println(rollno+" "+name+" "+college);}
11. public static void main(String args[]){
12. Student s1 = new Student8(111,"Karan");
13. Student  s2 = new Student8(222,"Aryan");
15. s1.display();
16. s2.display();
17. }
18. }

Unchecked Exception in Java is those Exceptions whose handling is NOT verified during Compile time . These exceptions occurs because of bad programming. The program won't give a compilation error. All Unchecked exceptions are direct sub classes of RuntimeException class.

Exception which are checked at Compile time called Checked Exception. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throwskeyword.

1. Primitive—which include Integer, Character, Boolean, and Floating Point.
2. Non-primitive—which include Classes, Interfaces, and Arrays

In Java we use final keyword to variables to make its values not to be changed.

class ChessAlgorithm {

enum ChessPlayer { WHITE, BLACK }

...

**final** ChessPlayer getFirstPlayer() {

return ChessPlayer.WHITE;

}

...

}

Main method:

 when you execute a class with the Java interpreter, the runtime system starts by calling the class's main() method. The main() method then calls all the other methods required to run your application.

import java.util.Date;

class DateApp {

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

Date today = new Date();

System.out.println(today);

**}**

}

Variable:

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type.

Abstract class:

1. **abstract** **class** Bike{
2. **abstract** **void** run();
3. }
4. **class** Honda4 **extends** Bike{
5. **void** run(){System.out.println("running safely..");}
6. **public** **static** **void** main(String args[]){
7. Bike obj = **new** Honda4();
8. obj.run();
9. }
10. }

Interface:

|  |
| --- |
| interface Animal {  public void eat();  public void travel();  } /Implementing the above interface |
| public class MammalInt implements Animal {  public void eat() {  System.out.println("Mammal eats");  }  public void travel() {  System.out.println("Mammal travels");  }  public int noOfLegs() {  return 0;  }  public static void main(String args[]) {  MammalInt m = new MammalInt();  m.eat();  m.travel();  }  }   * Add elements to hash map   public class Player {    public Player() {  }   public void setScore(int score) {      this.score = score;  }   public void setName(String name) {      this.name = name;  }   private String name;  private int score;   public Player(String name, int score) {      this.name = name;      this.score = score;  }   public String getName() {       return name;   }   @Override  public String toString() {      return "Player{" + "name=" + name + "score=" + score + '}';  }   public int getScore() {      return score;  }  Array list:  public class ArrayListDemo {  public static void main(String args[]) {  // create an array list  ArrayList al = new ArrayList();  System.out.println("Initial size of al: " + al.size());  // add elements to the array list  al.add("C");  al.add("A");  al.add("E");  al.add("B");  al.add("D");  al.add("F");  al.add(1, "A2");  System.out.println("Size of al after additions: " + al.size());  // display the array list  System.out.println("Contents of al: " + al);  // Remove elements from the array list  al.remove("F");  al.remove(2);  System.out.println("Size of al after deletions: " + al.size());  System.out.println("Contents of al: " + al);  }  }  Hashset:  import java.util.\*;  public class HashSetDemo {  public static void main(String args[]) {  // create a hash set  HashSet hs = new HashSet();    // add elements to the hash set  hs.add("B");  hs.add("A");  hs.add("D");  hs.add("E");  hs.add("C");  hs.add("F");  System.out.println(hs);  }  } |

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);

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] + ", ");

} } }

Jdbc

String selectTableSQL = "SELECT USER\_ID, USERNAME from DBUSER";

Statement statement = dbConnection.createStatement();

ResultSet rs = statement.executeQuery(selectTableSQL);

while (rs.next()) {

String userid = rs.getString("USER\_ID");

String username = rs.getString("USERNAME");

}

Try catch finally

public class TryExample {

public static void main(String[] args)

{

try {

int[] i = {1, 2, 3};

int x = i[3];//Change to 2 to see "return" result

return;

} catch (ArrayIndexOutOfBoundsException e) {

System.exit(0);

} finally {

System.out.println("finally");

}

}

|  |
| --- |
| implement method overloading ?  class MyClass {  int height;  MyClass() {  System.out.println("bricks");  height = 0;  }  MyClass(int i) {  System.out.println("Building new House that is " + i + " feet tall");  height = i;  }  void info() {  System.out.println("House is " + height + " feet tall");  }  void info(String s) {  System.out.println(s + ": House is " + height + " feet tall");  }  }  public class MainClass {  public static void main(String[] args) {  MyClass t = new MyClass(0);  t.info();  t.info("overloaded method");    //Overloaded constructor:  new MyClass();  }  } |
|  |
| implement method overriding ?  class Man{  public void eat()  {  System.out.println("Human is eating");  }  }  class Boy extends Man{  public void eat(){  System.out.println("Boy is eating");  }  public static void main( String args[]) {  Boy obj = new Boy();  obj.eat();  }  }  Retrieve values from 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);          }      }  }  Retrieve values from 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());          }        }    }  Workspace: |

File -> Switch Workspace -> Other... and type in my new workspace name.

New project:

1. Install the Eclipse IDE for Java Developers. ...
2. Click "File" → "New" → "Java Project". ...
3. Give the project a name. ...
4. Select the location for the project files. ...
5. Select Java Runtime Environment (JRE) you want to use. ...
6. Select your project layout. ...
7. Click "Next" to open the "Java Settings" window.

Create .java class

javac filename.java

|  |
| --- |
| creating method with return data type, we can return int/string/double/float/date etc |
| yes we can return a value from any of the specific method |
| 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?  Hard coding is not only part of java. It actually comes when we put the original variables and data values  In place of hard coding |
|  |
| Create default/paramterzied constructors? |
| when we don't declare a constructor ,, then a default constructor is created ,, when an object is declared and the values are passed in it then it is called as a paramterized constructor |

|  |
| --- |
| calling method with no return and parameter ?  public double calculateAnswer(double wingSpan, int numberOfEngines,  double length, double grossTons) {  Volume == length\* grossTons  } |
|  |
| Calling method with return and no parameter ?    Return type<----int total( int aNumber) {  int a\_value =aNumber+10;  return a\_value;  } |