# Program No.11: Code, execute and debug programs that uses array concept.

1. **Java Program to illustrate how to declare, instantiate, initialize and traverse the Java array. class** OneDimArray

{

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

{

}

}

**Output:**

**int** a[]=**new int**[5];//declaration and instantiation a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//traversing array System.*out*.println("Elements of array are");

**for**(**int** i=0;i<a.length;i++) //length is the property of array System.*out*.println(a[i]);

**Elements of array are 10**

**20**

**70**

**40**

**50**

1. **Java Program to illustrate the use of multidimensional array class** MultiDimArray

{

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

{

**int** arr[][]={{1,2,3},{2,4,5},{4,4,5}}; //declaring and initializing 2D array

//printing 2D array System.*out*.println("Elements of 2D array are"); **for**(**int** i=0;i<3;i++)

{

**for**(**int** j=0;j<3;j++)

{

System.out.println(arr[i][j]+" ");

}

System.out.println();

}

}

}

**Output:**

**Elements of 2D array are 1**

**2**

**3**

**2**

**4**

**5**

**4**

**4**

**5**

# Program No.12: Code, execute and debug programs to perform string manipulation.

**import** java.lang.String;

**class** StringDemo

{

**public static void** main(String arg[])

{

String s1 = **new** String("gpt mudhol"); String s2 = "GPT MUDHOL";

System.*out*.println("The string s1 is : " + s1); System.*out*.println("The string s2 is : " + s2); System.*out*.println("Length of the string s1 is : " + s1.length()); System.*out*.println("Length of the string s2 is : " + s2.length());

System.*out*.println("The String s1 in Upper Case : " + s1.toUpperCase()); System.*out*.println("The String s2 in Lower Case : " + s2.toLowerCase()); System.*out*.println("The first occurrence of a is at the position : "+ s1.indexOf('m')); System.*out*.println("s1 equals to s2 : " + s1.equals(s2));

System.*out*.println("s1 equals ignore case to s2 : " + s1.equalsIgnoreCase(s2)); System.*out*.println("Character at an index of 6 is :" + s1.charAt(6));

String s3 = s1.substring(4, 8); System.*out*.println("Extracted substring is :" + s3);

System.*out*.println("After Replacing m with b in s1 : "+ s1.replace('m', 'b')); System.*out*.println("After string concat :" + s1.concat(" Karnataka")); String s4 = " This is a book "; //White space before This word.

System.*out*.println("The string s4 is :" + s4); System.*out*.println("After string trim :" + s4.trim()); **int** result = s1.compareTo(s2); System.*out*.println("After compareTo");

**if** (result == 0)

System.*out*.println(s1 + " is equal to " + s2);

**else if** (result > 0)

System.*out*.println(s1 + " is greater than " + s2);

**else**

}

}

System.*out*.println(s1 + " is smaller than " + s2);

**Output:**

The string s1 is : gpt mudhol

The string s2 is : GPT MUDHOL Length of the string s1 is : 10 Length of the string s2 is : 10

The String s1 in Upper Case : GPT MUDHOL The String s2 in Lower Case : gpt mudhol

The first occurrence of m is at the position : 4 s1 equals to s2 : false

s1 equals ignore case to s2 : true Character at an index of 6 is :d Extracted substring is :mudh

After Replacing m with b in s1 : gpt budhol After string concat :gpt mudhol Karnataka The string s4 is : This is a book

After string trim :This is a book After compareTo

gpt mudhol is greater than GPT MUDHOL

# Program No.13: Code, execute and debug a program that implements the concept of inheritance.

**class** Room

{

**int** length,breadth; Room(**int** x, **int** y)

{

length = x; breadth = y;

}

**int** area()

{

**return** (length \* breadth);

}

}

**class** ClassRoom **extends** Room

{

**int** height;

ClassRoom(**int** x, **int** y, **int** z)

{

**super**(x, y); height = z;

}

**int** volume()

{

**return** (length \* breadth \* height);

}

}

**class** SubClass

{

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

{

ClassRoom cr = **new** ClassRoom(20, 30, 10);

**int** area = cr.area();

**int** volume =cr.volume();

}

}

**Output**

System.*out*.println("Area=" + area); System.*out*.println("Volume=" + volume);

**Area = 600**

**Volume = 6000**

# Program No.14: Design a class & implement like file parser and check compliance with OCP.

class Cuboid

{

public double length; public double breadth; public double height;

}

class Application

{

public double get\_total\_volume(Cuboid geo\_objects[])

{

double vol\_sum = 0;

for (Cuboid geo\_obj : geo\_objects)

{

vol\_sum += geo\_obj.length \* geo\_obj.breadth \* geo\_obj.height;

}

return vol\_sum;

}

}

public class OCP

{

public static void main(String args[])

{

Cuboid cb1 = new Cuboid(); cb1.length = 5;

cb1.breadth = 10;

cb1.height = 15;

Cuboid cb2 = new Cuboid(); cb2.length = 2;

cb2.breadth = 4;

cb2.height = 6;

Cuboid cb3 = new Cuboid(); cb3.length = 3;

cb3.breadth = 12;

cb3.height = 15;

Cuboid c\_arr[] = new Cuboid[3]; c\_arr[0] = cb1;

c\_arr[1] = cb2; c\_arr[2] = cb3;

Application app = new Application ();

double volume = app.get\_total\_volume(c\_arr); System.out.println ("The total volume is " + volume);

}

}

**Output:**

**The total volume is 1338.0**

**Program No.15: Code, execute and debug programs that uses**

1. **static binding**
2. **dynamic binding**
3. **Static binding**

**class** Dog

{

**private void** eat()

{

System.*out*.println("Dog is eating...");

}

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

{

Dog d1=**new** Dog(); d1.eat();

}

}

**Output:**

**Dog is eating...**

# Dynamic binding

**class** Animal

{

**void** eat()

{

System.*out*.println("animal is eating...");

}

}

**class** Dog1 **extends** Animal

{

**void** eat()

{

System.*out*.println("dog is eating...");

}

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

{

}

**Output:**

Animal a=**new** Dog1(); a.eat();

}

**Dog is eating...**

# Program No.16: Code, execute and debug program that uses abstract class to achieve abstraction.

**abstract class** Shape

{

**abstract void** draw();

}

//In real scenario, implementation is provided by others i.e. unknown by end user

**class** Rectangle **extends** Shape

{

**void** draw()

{

System.out.println("drawing rectangle");

}

}

**class** Circle **extends** Shape

{

**void** draw()

{

System.out.println("drawing circle");

}

}

//In real scenario, method is called by programmer or user

**class** TestAbstraction

{

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

{

Shape s=**new** Circle();

//In a real scenario, object is provided through method, e.g., getShape() method s.draw();

}

}

**Output:** drawing circle

# Program No.17: Code, execute and debug program that uses interface to achieve abstraction.

**interface** Area

{

**final static float** *pi* = 3.142F;

**float** compute(**float** x, **float** y);

}

**class** Rectangle **implements** Area

{

**public float** compute(**float** x, **float** y)

{

**return** ( x \* y);

}

}

**class** Circle **implements** Area

{

**public float** compute(**float** x, **float** y)

{

**return** (*pi* \* x \* x);

}

}

**class** InterfaceTest

{

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

{

Rectangle rect = **new** Rectangle(); Circle cir = **new** Circle();

Area area; area= rect;

System.*out*.println("Area of Rectangle = " + area.compute(10, 20)); area = cir;

System.*out*.println("Area of Circle = " + area.compute(30, 0));

}

}

# Output:

Area of Rectangle = 200 Area of Circle =3070.8

# Program No.18: Code, execute and debug program to read the content of the file and write the content to another file.

**(First create one text file- inputFile.txt and another text file outputFile.txt in C:drive\test folder ) import** java.io.File;

**import** java.io.FileInputStream; **import** java.io.FileWriter; **import** java.io.IOException; **import** java.util.Scanner;

**class** CopyContent

{

**public static void** main(String[] args) **throws** IOException

{

File file = **new** File("C:\\test\\inputFile.txt"); FileInputStream inputStream = **new** FileInputStream(file); Scanner sc = **new** Scanner(inputStream);

StringBuffer buffer = **new** StringBuffer();

**while**(sc.hasNext())

{

buffer.append(" "+sc.nextLine());

}

System.*out*.println("Contents of the file: "+buffer); File dest = **new** File("C:\\test\\outputFile.txt"); FileWriter writer = **new** FileWriter(dest); writer.write(buffer.toString());

writer.close();

System.*out*.println("File copied successfully ");

}

}

# Output:

Contents of the file: Welcome to GPT Mudhol This is example for checked exceptions. It uses throws keyword. Welcome to CS dept

File copied successfully.......

# Program No.19: Code, execute and debug program that handles checked and unchecked exceptions

**a) Checked Exceptions:**

import java.io.\*; class Checked

{

public static void main(String[] args)

{

}

}

**Output:**

FileReader file = new FileReader("C:\\test\\gpta.txt"); BufferedReader fileInput = new BufferedReader(file); for (int counter = 0; counter < 3; counter++) System.out.println(fileInput.readLine()); fileInput.close();



* To fix the above program, we either need to specify a list of exceptions using **throws**, or we need to use a **try-catch block.** We have used throws in the below program. Since *FileNotFoundException* is a subclass of *IOException*, we can just specify *IOException* in the throws list and make the above program compiler-error-free.

import java.io.\*; class Checked

{

public static void main(String[] args) **throws IOException**

{

FileReader file = new FileReader("C:\\test\\gpta.txt"); BufferedReader fileInput = new BufferedReader(file); for (int counter = 0; counter < 3; counter++) System.out.println(fileInput.readLine()); fileInput.close();

}

}

**Output:**

Welcome to GPT Athani

This is example for checked exceptions. It uses throws keyword.

**a) Unchecked Exceptions:**

class Unchecked

{

public static void main(String args[])

{

// Here we are dividing by 0 which will not be caught at compile time

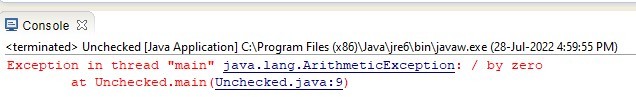
// as there is no mistake but caught at runtime because it is mathematically incorrect int x = 0;

int y = 10; int z = y / x;

}

}

**Output:**



# Program No.20: Code, execute and debug program to illustrate throwing our own exceptions or user defined exceptions.

import java.lang.Exception;

class MyException extends Exception

{

MyException(String message)

{

super(message);

}

}

class TestMyException

{

public static void main(String args[])

{

int x=5,y=1000; try

{

float z=(float) x/(float) y; if(z < 0.01)

{

throw new MyException(“Number is too small”);

}

}

catch(MyException e)

{

System.out.println(“Caught my exception”); System.out.println(e.getMessage());

}

finally

{

System.out.println(“I am always here”);

}

}

}

**Output:**

Caught my exception Number is too small I am always here

# Program No.21: Design an interface & implement it like one that builds different types of toys and check compliance with ISP.

interface Toy

{

void setPrice(double price); void setColor(String color);

}

interface Movable

{

void move();

}

interface Flyable

{

void fly();

}

class ToyHouse implements Toy

{

double price; String color; @Override

public void setPrice(double price)

{

this.price = price;

}

@Override

public void setColor(String color)

{

this.color=color;

}

@Override

public String toString()

{

return "ToyHouse: Toy house- Price: "+price+" Color: "+color;

}

}

class ToyCar implements Toy, Movable

{

double price; String color; @Override

public void setPrice(double price)

{

this.price = price;

}

@Override

public void setColor(String color)

{

this.color=color;

}

@Override

public void move()

{

System.out.println("ToyCar: Start moving car.");

}

@Override

public String toString()

{

return "ToyCar: Moveable Toy car- Price: "+price+" Color: "+color;

}

}

class ToyPlane implements Toy, Movable, Flyable

{

double price; String color; @Override

public void setPrice(double price)

{

this.price = price;

}

@Override

public void setColor(String color)

{

this.color=color;

}

@Override

public void move()

{

System.out.println("ToyPlane: Start moving plane.");

}

@Override public void fly()

{

System.out.println("ToyPlane: Start flying plane.");

}

@Override

public String toString()

{

return ("ToyPlane: Moveable and flyable toy plane- Price: "+price+"Color: "+color);

}

}

class ToyBuilder

{

public static ToyHouse buildToyHouse()

{

ToyHouse toyHouse=new ToyHouse(); toyHouse.setPrice(15.00); toyHouse.setColor("green");

return toyHouse;

}

public static ToyCar buildToyCar()

{

ToyCar toyCar=new ToyCar(); toyCar.setPrice(25.00); toyCar.setColor("red"); toyCar.move();

return toyCar;

}

public static ToyPlane buildToyPlane()

{

ToyPlane toyPlane=new ToyPlane(); toyPlane.setPrice(125.00); toyPlane.setColor("white"); toyPlane.move();

toyPlane.fly(); return toyPlane;

}

}

public class ToyISPTest

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

ToyHouse toyHouse=ToyBuilder.buildToyHouse(); System.out.println(toyHouse);

ToyCar toyCar=ToyBuilder.buildToyCar();; System.out.println(toyCar);

ToyPlane toyPlane=ToyBuilder.buildToyPlane(); System.out.println(toyPlane);

}

}

**Output:**

ToyHouse: Toy house- Price: 15.0 Color: green ToyCar: Start moving car.

ToyCar: Moveable Toy car- Price: 25.0 Color: red ToyPlane: Start moving plane.

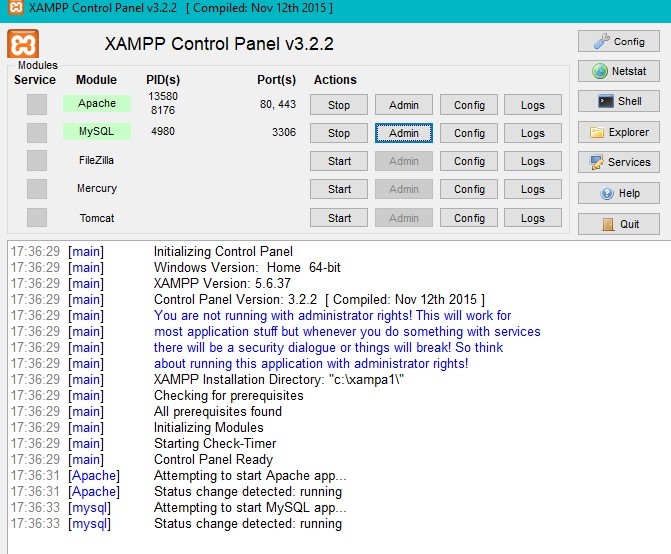
ToyPlane: Start flying plane.

ToyPlane: Moveable and flyable toy plane- Price: 125.0 Color: white

# Program No.22: Code, execute and debug programs to connect to database through JDBC and perform basic DB operations.

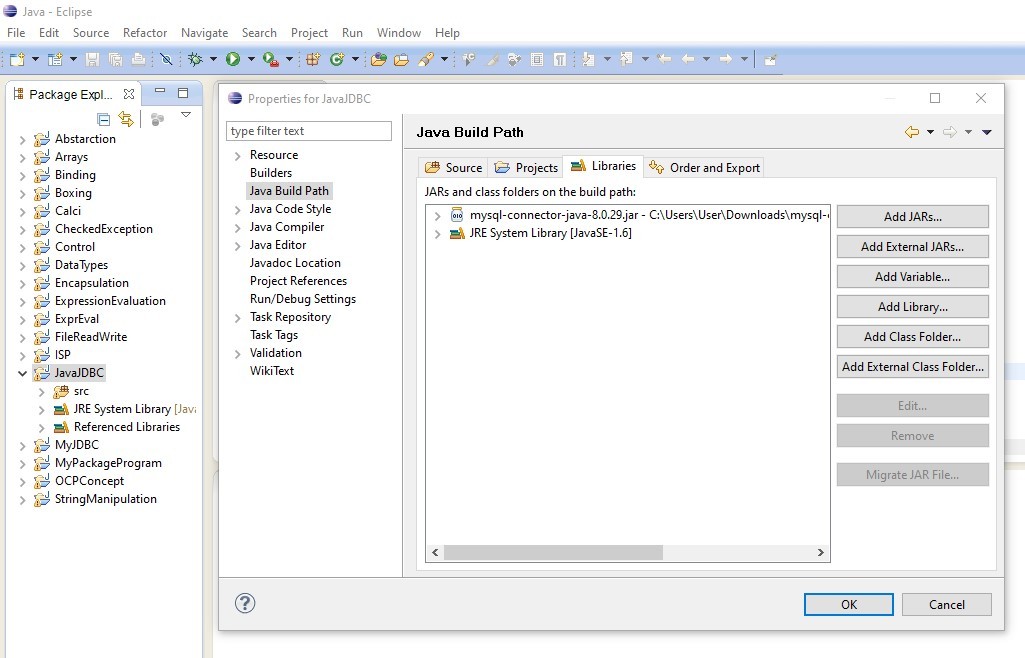
**Step 1:** In addition to JDK and Eclipse environment, install Xampp software for Apache server and MySql service.

**Step 2:** Now open Xampp control panel to start Apache and MySql services as shown below. Then click on MySql-Admin button to open MySql <http://localhost/phpmyadmin/> in brower.

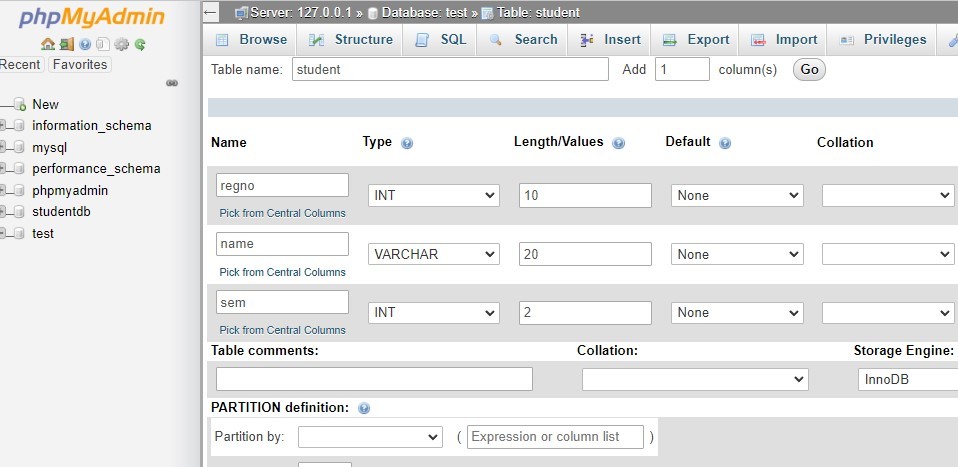


**Step 3:** To connect MySql databse in Java using Eclipse, follow below steps.

* Open Eclipse IDE and create new Java project named JavaJDBC and click finish.
* Create a new Java class with DBTest and click on the finish button.
* In order to connect Java program (DBTest.java) with MySQL database, we need to download and include MySQL JDBC driver which is a JAR file, namely **mysql-connector-java-8.0.29.jar.**
* Now right click on JavaJDBC project to include connector and go to properties.
* Click on Java build path option-> click on libraries and then click on Add External JARS.
* Now select downloaded jar file **mysql-connector-java-8.0.29.jar.** & click open.
* Click on OK and close.



**Step 4:** Now in browser go to myphpadmin page and create student table in test database with following fields as shown below and click save.



**Connecting Java Program with MySQL Database**

* After adding jar file, connect the Java program with MySQL Database.
  1. Establish a connection using DriverManager.getConnection(String URL) and it returns a Connection reference.
  2. In String URL parameter write like this :

**jdbc:mysql://localhost:3306/test”, “root”, “password”**

Where,

* + jdbc is the API.
  + mysql is the database.
  + localhost is the name of the server in which MySQL is running.
  + 3306 is the port number.
  + test is the database name. If the database name is different, then replace this name with the correct database name.
  + root is the username of the MySQL database. It is the default username for the MySQL database.
  + password is the password that is given while installing the MySQL database.
* SQL Exception might occur while connecting to the database, try-catch block must be used.

**Step 5:** Write below code in DBTest class Eclipse environment.

**import** java.sql.\*;

**public class** DBTest

{

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

{

String url= "jdbc:mysql://localhost:3306/test"; // table URL String uname = "root"; // MySQL credentials

String pw = "";

**try**

{

//Loading MySQL Driver Class.*forName*("com.mysql.cj.jdbc.Driver");

// Establishing connection with MySQL

Connection con = DriverManager.*getConnection*(url,uname,pw); System.*out*.println("Java Connection to MySQL Established successfully");

// Creating Statement object for query execution Statement st=con.createStatement();

// Delete the table student if already present in the test database String deltbl= "DROP TABLE STUDENT"; st.executeUpdate(deltbl);

// Create a table STUDENT in database test

String qrytbl= "CREATE TABLE STUDENT(regno int,name varchar(30),sem int)"; st.executeUpdate(qrytbl);

// Insert values into the STUDENT table

String qry1="INSERT INTO STUDENT values(2001,'Anand',4)"; st.executeUpdate(qry1);

String qry2="INSERT INTO STUDENT values(2002,'Santosh',4)"; st.executeUpdate(qry2);

String qry3="INSERT INTO STUDENT values(2003,'Ullas',4)"; st.executeUpdate(qry3);

System.*out*.println("Table Values insertion is successful");

// Query to retrieve values from table

String query= "SELECT \* FROM STUDENT"; ResultSet rs = st.executeQuery(query);//Execute query **while** (rs.next())

{

//Retrieve row-wise values of regno, name and sem columns

**int** regno = rs.getInt("regno"); String name= rs.getString("name"); **int** sem=rs.getInt("sem");

// Display the result on console System.*out*.println(regno + " " + name+ " "+ sem);

}

st.close(); // close statement con.close(); // close connection

System.*out*.println("MySQL Connection Closed successfully!");

}

**catch**(Exception e)

{

System.*out*.println("Error while executing program:" + e);

}

}

}

**Output:**

