1)Q1 : Write a Java program to create a new array list, add some elements (string) and print out the collection by using for-each loop

import java.util.\*;

import java.util.Arraylist;

class Student

{

public static void main(String args[])

{

ArrayList<String> list = new Arrraylist<String>();

list.add(e:"Green");

list.add(e:"Violet");

list.add(e:"Orange");

list.add(e:"White");

list.add(e:"Black");

for(String color:list);

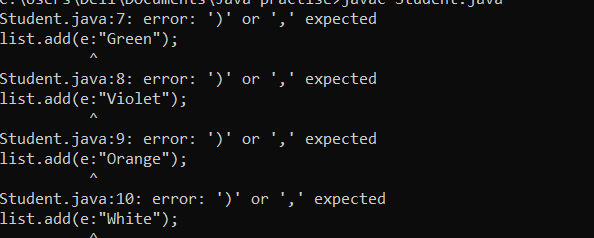
{

System.out.println("color");

}

}

}



1)

class Student

{

public static void main(String args[])

{

ArrayList<String> list = new Arrraylist<String>();

list.add("Green");

list.add("Violet");

list.add("Orange");

list.add("White");

list.add("Black");

for(int i = 0; i < list.get(i);i++);

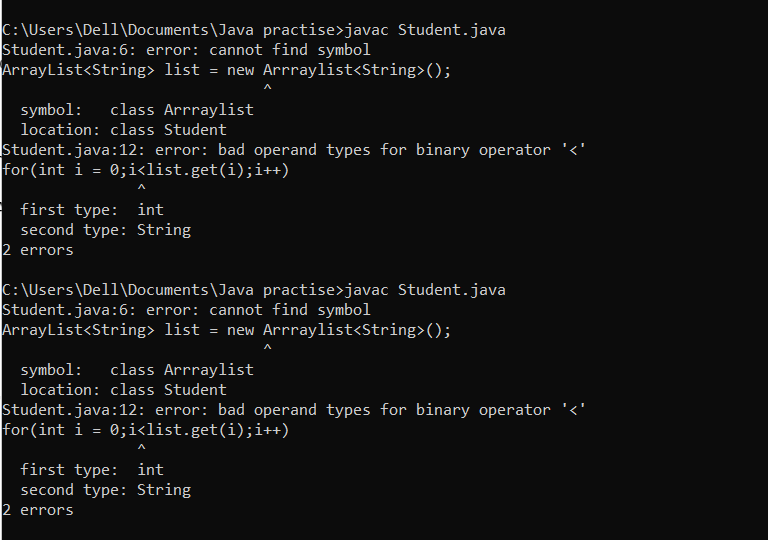
{

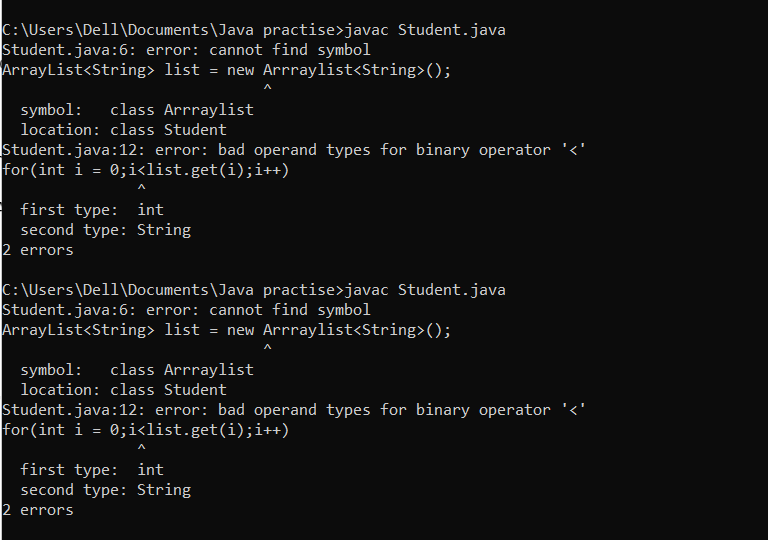
System.out.println("list.get(i)");

}

}

}





Q3 : Write a program to create a class named shape. In this class we have three

sub classes circle, triangle and square, each class has two member function

named draw () and erase (). Create these using Runtime Polymorphism concepts

class Shape{

void draw(){

System.out.println("Drawing Shape");

}

void erase(){

System.out.println("Erasing Shape");

}

}

class Circle extends Shape{

@override

void draw(){

System.out.println("Drawing Circle");

}

@Override

void erase(){

System.out.println("Erasing Circle");

}

}

class Triangle extends Shape{

@Overridevoid draw(){

System.out.println("Drawing Triangle");

}

}

@Override

void erase(){

System.out.println("Erasing Triangle");

}

}

class Square extends Shape{

@Override

void draw(){

System.out.println("Erasing Square");

}

}

public class Solution{

public static void main(String[] args){

//TODO Autogenerated method stub

Shape c = new Circle();

Shape t = new Triangle();

Shape s = new Square();

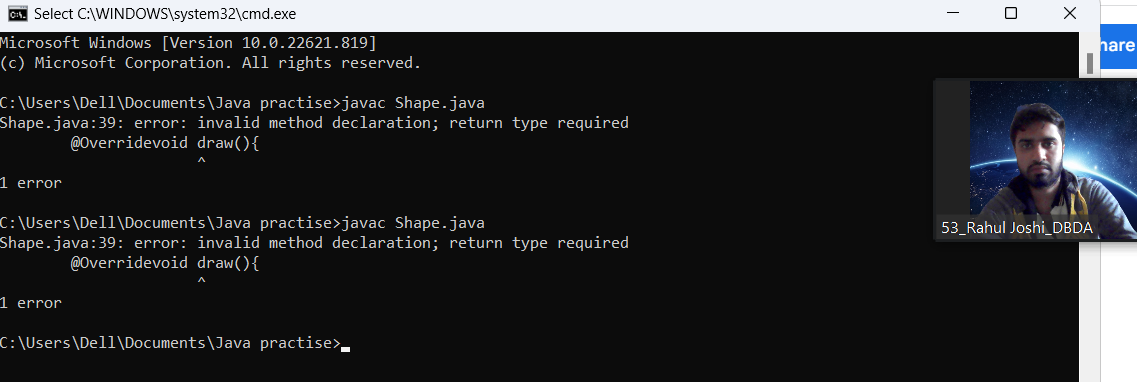
c.draw();c.erase();

t.draw();t.erase();

s.draw();s.erase();

}

}



Q2 : Develop a class BankAccount having following data members : (10 Marks)

int accno

double balance

Write appropriate constructors to initialize data members

Define the following functions :

withdraw : balance will reduce

deposit : balance will increase

show : display accno and balance

If user tries to withdraw more than the balance, use exception handling code. Demonstrate the concept of exception handling in main() function

import java.util.Scanner;

class BankAccount

{

int accno;

double balance;

public BankAccount

{

this.accno=accno;

this.balance=balance;

}

public void withdraw(double amount)

{

try

{

System.out.println("Invalid amount entered");

}

catch(Exception e)

{

System.out.println("Error:withdraw amount should be less than balance");

}

}

else

{

balance=balance-amount;

System.out.println("balance of account after withdraw is:"+balance);

}

}

public void deposit(double amount)

{

balance=balance+amount

{

System.out.println("balance of account after depositing amount is:"+balance);

}

public void show\_account\_details()

{

System.out.println("account number:+accno");

System.out.println("balance of account:+balance")

}

}

public class bank2

{

public static void(String args[])

{

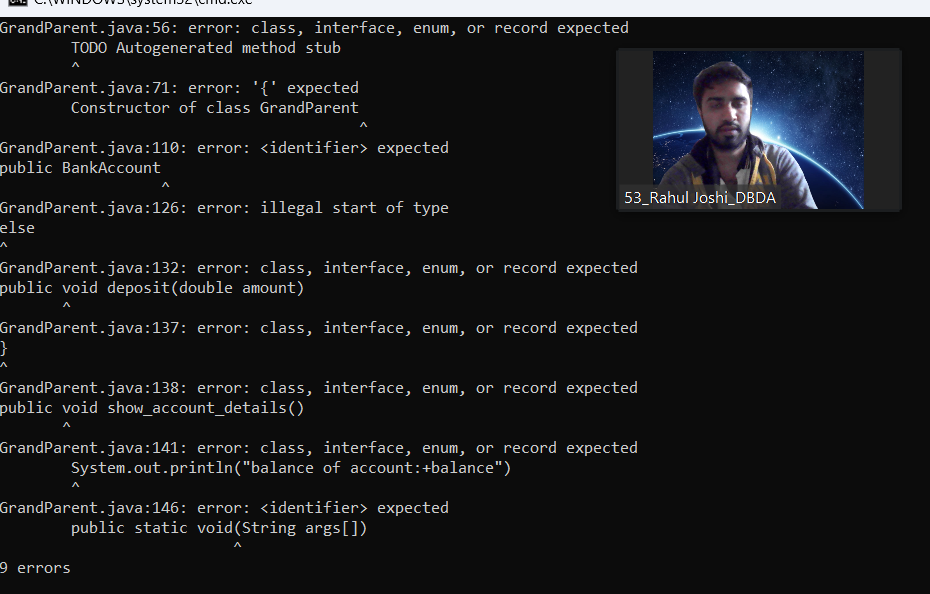
BankAccount ba = new BankAccount(123,100000);

ba.withdraw(12000);

ba.deposit(120000);

}

}



4)

class GrandParent{

String grandFathername;

String grandMothername;

//Constructor of class GrandParent

GrandParent(String GFN,String GMN){

this.grandFatherName=GFN;

this.grandMotherName=GMN;

System.out.println("Grand parent names are:");

System.out.println("GrandFather:"+this.grandFatherName);

System.out.println("Grandmother:" +this.grandMotherName);

}

}

class Parent extends GrandParent{

String fatherName;

String motherName;

{

super(grandFatherName,grandMotherName);

this.FatherName=FatherName;

this.MotherName=MotherName;

}

public Parent(String grandFatherName,String GrandMotherName){

super(grandFatherName,grandMotherName);

}

{

public static void main(String args[])

{

child c = new child("Rahul","Sharvari","Pratiksha","Aniket");

}

}

}

