PROJECT REPORT

The Amusement park

*The fun starts here*

Name :- Rakshith Alaham Gangeya

Registration No: 189302117

Section :- CSE-D

Roll-No :- 51

The Amusement park

The fun starts here

# introduction

This is a sample program for managing and running the rides in an Amusement park. This program has two approaches

1. Admin Approach
2. Guest Approach

The Admin approach focuses on the Database Management and the ability to control and monitor rides in an Amusement park

The Guest Approach enables the guests the create an ID which further enables them to access rides depending on the age restriction of the rides and the availability.

The main aim of this project is to implement all the concepts of JAVA which is focused upon during the duration of our course.

# current work

The program is designed to handle the free flow of customers entering an Amusement park by providing basic details of the rides available in the Amusement park and by allowing them to choose the rides they wish to take.

Different databases have been created for Customer details and the Log of each ride in the Amusement park which contains the list of people who have taken a specific ride.

Admin access has been created with the help of an interface to maintain abstraction and the program enables the admin user to control all the rides in the amusement park and has view and edit access to all the databases relating to the amusement park.

# future work

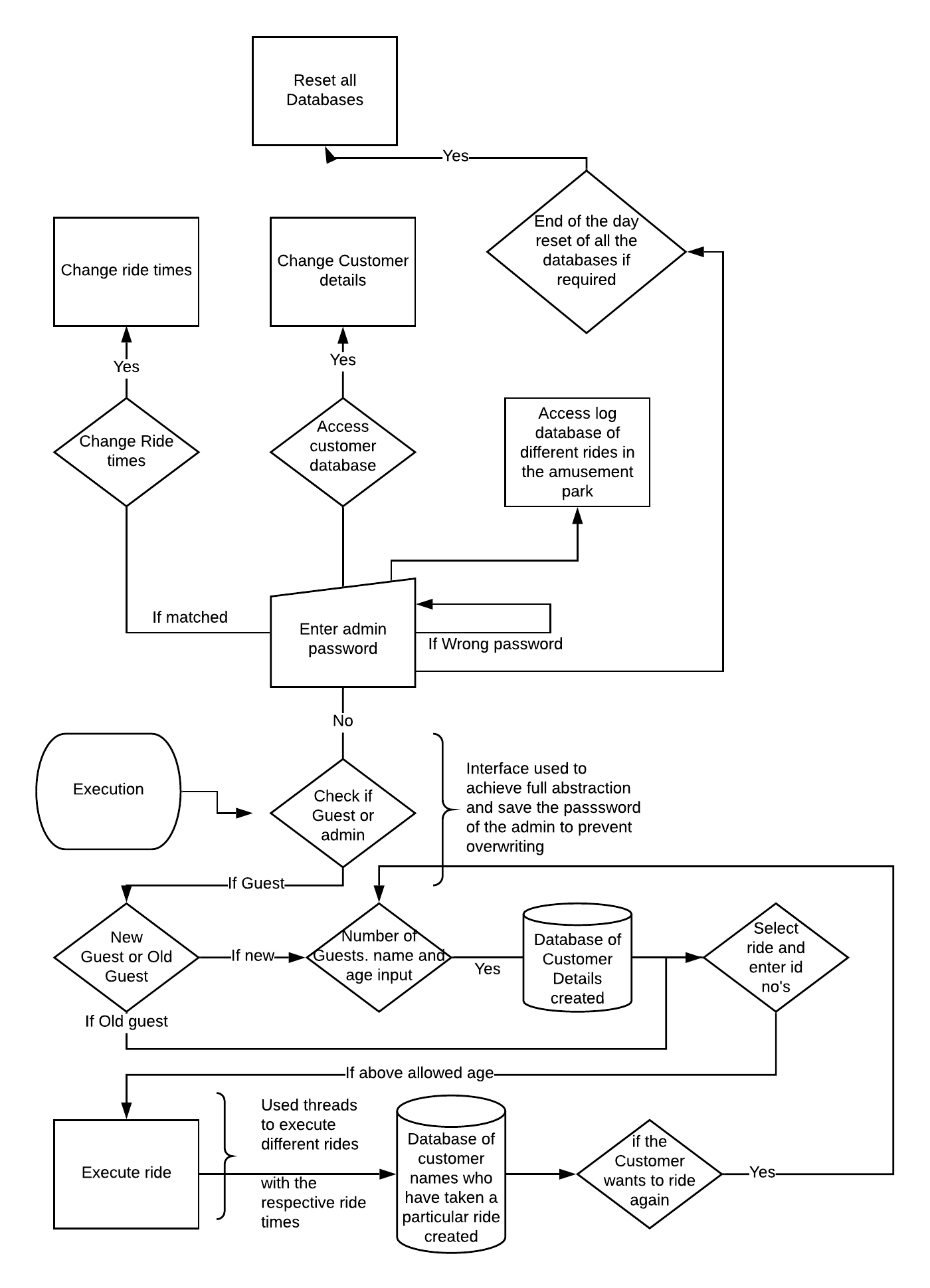
We plan to implement Hotel Room booking facility for the customers if they are interested in the same.

We also plan on implementing a feedback system which accepts valuable data from customers regarding how the Amusement park can be improved in different ways possible.

# Software requirements

1. JDK 8.0 or above installed on Windows/ Ubuntu/MacOS
2. Command prompt/terminal with environment variables added to it.

# flow diagram and methodology



# Topics used and explanation

1. Abstract class:

**This is used to create a blueprint of the customer details so that the main Details class cannot have any memory of its own and only it can be used as a reference class for creating objects of customers in the rides class.**

1. Packages:

**Two packages have been used**

1. **Detailspackage**
2. **Adminpackage**
3. Method Overriding:

**As we have used Abstract and interface classes, overriding is essential without which the methods in both abstract and interface won’t be of any use.**

**Runtime method overriding also implemented using abstract class as a reference class.**

1. Array of Objects:

**An array of objects has been created to hold the data of the customers coming into the Amusement park which is of great use to shorten the lines of code.**

1. Wrapper class:

**During file input to Database of customers we have used Integer.toString() method to convert a primitive int to a Wrapper class String for easier input to the Database.txt file.**

1. Interface:

**Interface has been used for admin class to achieve complete abstraction, so as to avoid changing of admin username and password during runtime by the user.**

**Also Interface Admin has been used in a package so that if in future upgrades of hotel booking, it can only be used when required.**

1. Constructor Overloading
2. Method overloading:

**Method overloading has been used in a case when the customer does not want to share his age then a constructor is called without the age parameter so create an object of that particular person.**

1. Input/Output and FileHandling:

**For creating databases of customers and log book of every single ride we have used FileOutputStream((“Database.txt”),true)) constructor for appending and input on top of the existing file of Database.txt**

**When the admin requests to display any of the databases we use FileInputStream and BufferedReader showing the result during prompt.**

**PrintWriter has been used to reset all the databases if the admins prompts to do so.**

1. Inheritance:

**We have used Inheritance to extend the abstract class Details and the interference Admin to the main class Bleh.**

1. MultiThreading :

**When a Guests wants to ride a ride in the Amusement park, distinct threads are created for every single person after which the start() method is used..**

**Also we have used join method in such cases when only one person can take a ride**

**Example: Only one person can ride a single chaired rollercoaster at once so when join() method is called after the start() method, the program waits until the ride of that particular person is completed before it moves on to the next person waiting for the ride or to implement the remaining part of the program.**

1. Exception Handling:
2. **We have used the Throws keyword in the main(String args[]) run method so that the program expects an IOException in case the input entered is wrong**
3. **Custom Exception has been created for InvalidAgeException and InvalidpasswordException . Invalid age occurs when the age of the guest is below the age limit of a particular ride. Invalid Password exception occurs when the admin enters the password wrong twice.**
4. **Throw keyword has been used to throw custom exception and to throw InterruptedException.**
5. **Try and catch have been used to handle all the above thrown exceptions**
6. **Finally method is used at the very end of the program for proper and smooth termintation.**
7. String class:

**String class static method have been used such as String.charAt(), String.length().**

# CODE

package detailspackage;

import java.util.\*;

import java.io.\*;

public abstract class Details extends Thread // Abstract Class used here

{

public static int count=0; // Static variable used here

public String name;

public int age;

public int id;

public int numofridesallowed=10;

public Details()

{

count++;

this.id=count;

}

public Details(String meh){System.out.println("Admin "+meh+" created");} // Constructor Overloading used here

public abstract void set\_info(String name,int age); // Abstract method used here

public abstract void show\_info();

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

{System.out.println("w");}

}

package adminpackage; //custom package created here

public interface Admin //interface used here

{

public static final String password="hello123";

}

import java.util.\*;

import java.io.\*;

import detailspackage.Details; // Custom package which includes an abstract class of the blueprint of customer details

import adminpackage.Admin;

class InvalidAgeExcpetion extends Exception //Custom Exception created

{

InvalidAgeExcpetion(String s){

super(s);

}

}

class InvalidPassword extends Exception //Custom Exception created

{

InvalidPassword(String s){

super(s);

}

}

class Rides extends Details implements Admin //Outer Class used here

{ public Rides(String meh){System.out.println("Admin "+meh+" created");}

Rides(){}

int time = 3000;

public void run()

{

try{

System.out.println(this.name+" started taking the ride");

Thread.sleep(this.time);

this.numofridesallowed--;

System.out.println(this.name+" finished riding the ride");

}catch(InterruptedException e){System.out.println(e);}

}

class Rollercoaster //Innerclass used here

{ boolean state= true;

int minage=18;

int ridecount;

int rideguestid[]=new int[100];

}

class Bumpercars

{ boolean state= true;

int minage=12;

int ridecount;

int rideguestid[]=new int[100];

}

class Ferriswheel

{ boolean state= true;

int minage=10;

int ridecount;

int rideguestid[]=new int[100];

}

class Waterpark

{ boolean state= true;

int minage=7;

int ridecount;

int rideguestid[]=new int[100];

}

@Override

public void set\_info(String name,int age) // Method Overriding used here

{

this.name=name;

this.age=age;

}

public void set\_info(String name) // Method Overriding used here

{

this.name=name;

}

@Override

public void show\_info()

{

System.out.println("Name: "+name+" Age: "+age+" Id: "+id+" ");

}

}

class Bleh extends Rides implements Admin

{

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

{ int t=0;

Boolean temp2=false;

Rides d[]=new Rides[100];

Admin moderator=new Rides("Batman");

count--;

Scanner sc=new Scanner(System.in); // Array of objects used here

d[0]=new Rides();

System.out.println(" \*\*\*\*\*\*WELCOME TO THE AMUSMENT PARK\*\*\*\*\*\*\*\*\*\*\* ");

Rides.Rollercoaster rollcoast = d[0].new Rollercoaster(); // Objects of Innerclass Created

Rides.Bumpercars bumpcars = d[0].new Bumpercars();

Rides.Ferriswheel wheel= d[0].new Ferriswheel();

Rides.Waterpark watergame = d[0].new Waterpark();

while(true){

System.out.println("1.Guest\n2.Admin\n3.exit");

System.out.println("Enter Input");

int identity= sc.nextInt();

if(identity==1){

System.out.println("1.NewGuest\n2.Oldguest");

System.out.println("Enter Input");

int temp1=sc.nextInt();

if(temp1==1){

try{

FileOutputStream fout=new FileOutputStream(("Database.txt"),true);

System.out.println("Enter the number of guests");

int numofguests=sc.nextInt();

for(int i=1;i<=numofguests;i++)

{ if(temp2==true)

d[count]=new Rides();

temp2=true;

sc.nextLine();

System.out.println("Enter name and age");

String tempname=sc.next();

int tempage=sc.nextInt();

d[count-1].set\_info(tempname,tempage);

String strage= Integer.toString(tempage); // Wrapper class used here

String addedstrings=(tempname+"-"+strage+"-"+(count));

for(int j=0;j<addedstrings.length();j++)// Name and age of all pushed in to a database txt file using FileOutputStream

{

fout.write(addedstrings.charAt(j));

}

fout.write('\n');

}

fout.close();

}catch(Exception e){System.out.println(e);}

System.out.println("Details of all the guests are");

for(int i=0;i<count;i++)

{

d[i].show\_info();

}

System.out.println("Each of you are allowed to take 10 Rides from our amusment park");

}

System.out.println("1.rides\n");

int ch=sc.nextInt();

switch (ch)

{

case 1: System.out.println("Which Ride do you want to take?\n1.Rollercoaster\n2.Bumpercars\n3.Ferriswheel\n4.Waterpark");

System.out.println("Enter Input");

int choice=sc.nextInt();

System.out.println("Enter the number of guests taking the ride");

int ridenum=sc.nextInt();

int temp[]=new int[ridenum];

System.out.println("Enter the id's of the guests taking the ride");

for(int i=0;i<ridenum;i++)

{

temp[i]=sc.nextInt();

}

for(int i=0;i<ridenum;i++)

{

if(choice==1)

{ if(rollcoast.state==true){

if(d[temp[i]-1].age<rollcoast.minage)

{try{throw new InvalidAgeExcpetion("Guest "+d[temp[i]-1].name+"'s age is under permitted age limit"); //Custom Exception Thrown here

}catch(Exception e){System.out.println(e);}}

else

{

if(d[temp[i]-1].numofridesallowed>0)

{ try{

FileOutputStream fout= new FileOutputStream(("Rollercoaster.txt"),true);

for(int j=0;j<d[temp[i]-1].name.length();j++)

{

fout.write(d[temp[i]-1].name.charAt(j));

}

fout.write('\n');

fout.close();

}catch(Exception e){System.out.println(e);}

rollcoast.ridecount++;

d[temp[i]-1].start(); // Thread sleep function used here

try{

d[temp[i]-1].join(); // Thread join fucntion used here

}catch(Exception e){System.out.println(e);}

}

else

System.out.println("You have used the number of rides allowed");

}

}

else

{System.out.println("Ride is currently closed");

break;}

}

if(choice==2)

{ if(bumpcars.state==true){

if(d[temp[i]-1].age<bumpcars.minage)

{try{throw new InvalidAgeExcpetion("Guest "+d[temp[i]-1].name+"'s age is under permitted age limit");

}catch(Exception e){System.out.println(e);}}

else

{

if(d[temp[i]-1].numofridesallowed>0)

{

try{

FileOutputStream fout= new FileOutputStream(("Bumpercars.txt"),true);

for(int j=0;j<d[temp[i]-1].name.length();j++)

{

fout.write(d[temp[i]-1].name.charAt(j));

}

fout.write('\n');

fout.close();

}catch(Exception e){System.out.println(e);}

bumpcars.ridecount++;

d[temp[i]-1].start();

}

else

System.out.println("You have used the number of rides allowed");

}

}

else

{System.out.println("Ride is currently closed");

break;}

}

if(choice==3)

{ if(wheel.state==true){

if(d[temp[i]-1].age<wheel.minage)

{try{throw new InvalidAgeExcpetion("Guest "+d[temp[i]-1].name+"'s age is under permitted age limit");

}catch(Exception e){System.out.println(e);}}

else

{

if(d[temp[i]-1].numofridesallowed>0)

{

try{

FileOutputStream fout= new FileOutputStream(("Ferriswheel.txt"),true);

for(int j=0;j<d[temp[i]-1].name.length();j++)

{

fout.write(d[temp[i]-1].name.charAt(j));

}

fout.write('\n');

fout.close();

}catch(Exception e){System.out.println(e);}

wheel.ridecount++;

d[temp[i]-1].start(); // Thread sleep function used here

}

else

System.out.println("You have used the number of rides allowed");

}

}

else

{System.out.println("Ride is currently closed");

break;}

}

if(choice==4)

{ if(watergame.state==true){

if(d[temp[i]-1].age<watergame.minage)

{try{throw new InvalidAgeExcpetion("Guest "+d[temp[i]-1].name+"'s age is under permitted age limit"); // Custom Exception thrown here

}catch(Exception e){System.out.println(e);}}

else

{

if(d[temp[i]-1].numofridesallowed>0)

{

try{

FileOutputStream fout= new FileOutputStream(("Waterpark.txt"),true);

for(int j=0;j<d[temp[i]-1].name.length();j++)

{

fout.write(d[temp[i]-1].name.charAt(j));

}

fout.write('\n');

fout.close();

}catch(Exception e){System.out.println(e);}

watergame.ridecount++;

d[temp[i]-1].start(); // Thread sleep function used here

}

else

System.out.println("You have used the number of rides allowed");

}

}

}

}

break;

}

}

else if(identity==2)

{

System.out.println("Please Enter the admin password");

String passwordcheck=sc.next();

if(passwordcheck.equals(moderator.password)==true)

System.out.println("Welcome moderator");

else

{

System.out.println("Incorrect password please enter the password again");

passwordcheck=sc.next();

if(passwordcheck.equals(moderator.password)==true)

System.out.println("Welcome moderator");

else

{

try

{

throw new InvalidPassword("Entered wrong password twice");

}catch(Exception e){System.out.println(e);}

}

}

System.out.println("1.Manage Rides\n2.View Customer Database\n3.Edit ride time\n4.Rollercoaster Database\n5.Bumpercars Database\n6.Ferriswheel Database\n7. Waterpark Database\n8.Delete all databases");

int ch= sc.nextInt();

switch(ch)

{

case 1: System.out.println("Select the ride");

System.out.println("1.Rollercoaster\n2.Bumpercars\n3.Ferriswheel\n4.Waterpark");

System.out.println("Enter Input");

int ch1=sc.nextInt();

System.out.println("1.Turn on ride\n2.Turn off ride");

System.out.println("Enter Input");

ch=sc.nextInt();

if(ch==1)

{

if(ch1==1)

{

if(rollcoast.state==true)

System.out.println("Already on");

else

rollcoast.state=true;

}

if(ch1==2)

{

if(bumpcars.state==true)

System.out.println("Already on");

else

bumpcars.state=true;

}

if(ch1==3)

{

if(wheel.state==true)

System.out.println("Already on");

else

wheel.state=true;

}

if(ch1==4)

{

if(watergame.state==true)

System.out.println("Already on");

else

watergame.state=true;

}

}

if(ch==2)

{

if(ch1==1)

{

if(rollcoast.state==false)

System.out.println("Already off");

else

rollcoast.state=false;

}

if(ch1==2)

{

if(bumpcars.state==false)

System.out.println("Already on");

else

bumpcars.state=false;

}

if(ch1==3)

{

if(wheel.state==false)

System.out.println("Already on");

else

wheel.state=false;

}

if(ch1==4)

{

if(watergame.state==false)

System.out.println("Already on");

else

watergame.state=false;

}

}

break;

case 2: FileInputStream fstream = new FileInputStream("Database.txt"); // BufferedReader used here

BufferedReader br = new BufferedReader(new InputStreamReader(fstream));

String strLine;

System.out.println("Name-Age-ID");

while((strLine=br.readLine())!=null)

{

System.out.println(strLine);

}

fstream.close();

System.out.println("Total number of guests = "+(count));

break;

case 3: System.out.println("Enter the new ride time");

int ridetime= sc.nextInt();

d[0].time=ridetime;

break;

case 4: FileInputStream fstream1 = new FileInputStream("Rollercoaster.txt"); // BufferedReader used here

BufferedReader br1 = new BufferedReader(new InputStreamReader(fstream1));

System.out.println("Name");

while((strLine=br1.readLine())!=null)

{

System.out.println(strLine);

}

fstream1.close();

break;

case 5: FileInputStream fstream2 = new FileInputStream("Bumpercars.txt"); // BufferedReader used here

BufferedReader br2 = new BufferedReader(new InputStreamReader(fstream2));

System.out.println("Name");

while((strLine=br2.readLine())!=null)

{

System.out.println(strLine);

}

fstream2.close();

break;

case 6: FileInputStream fstream3 = new FileInputStream("Ferriswheel.txt"); // BufferedReader used here

BufferedReader br3 = new BufferedReader(new InputStreamReader(fstream3));

System.out.println("Name");

while((strLine=br3.readLine())!=null)

{

System.out.println(strLine);

}

fstream3.close();

break;

case 7: FileInputStream fstream4 = new FileInputStream("Waterpark.txt"); // BufferedReader used here

BufferedReader br4 = new BufferedReader(new InputStreamReader(fstream4));

System.out.println("Name");

while((strLine=br4.readLine())!=null)

{

System.out.println(strLine);

}

fstream4.close();

break;

case 8: PrintWriter pw = new PrintWriter("Database.txt");

PrintWriter pw1 = new PrintWriter("Ferriswheel.txt");

PrintWriter pw2 = new PrintWriter("Rollercoaster.txt");

PrintWriter pw3 = new PrintWriter("Bumpercars.txt");

PrintWriter pw4 = new PrintWriter("Waterpark.txt");

pw.close();

pw1.close();

pw2.close();

pw3.close();

pw4.close();

break;

default: System.out.println("Invalid Input");

}

}

else if(identity==3)

{

try{}

finally{ // Finally used here

System.out.println("Thank you for visiting THE AMUSMENT PARK");

}

}

}

}

@Override

public void set\_info(String name,int age) // Method Overriding used here

{

this.name=name;

this.age=age;

}

@Override

public void show\_info()

{

System.out.println("Name: "+name+"Age: "+age+"Id: "+id+" ");

}

}