CMP-4008Y Coursework 2 - Arcade System and Simulation

100473202 (bjj24bdu)

Mon, 5 May 2025 16:31

Contents

Simulation.java

```java
/*================================================


File                        :  Arcade.java

date                        :  14/4/2025

Author                      :  Benedict Ward

Description                 :  worth upto 25 marks, mainly linking up the types of
    arcade games
                               to customers

Possible Exceptions         :

History                     :  28/2/2025 v1.0 - made the static helper function
    readFromFile
                                                initialiseArcade now reads from both
                                                    given files
                                                11:09pm initialiseArcade now doesnt
                                                    throw any errors
                                                and when prompted gave mantis toboggan
                                                    as the richest customer
                                                11:18pm moved checking the gameType to a
                                                     switch case statement
                               1/3/2025 v1.01 - fully account for possible Exception

                               14/4/2025 v1.02 - better FileNotFoundException handling
================================================*/
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;



public final class Simulation {
    public static void main(String[] args){

        File customersFile = new File("customers.txt");
        File arcadeGamesFile = new File("games.txt");
        File transactionsFile = new File("transactions.txt");

        Arcade arcade = null;

        try {
            arcade = initialiseArcade("arcadeName", arcadeGamesFile, customersFile);
        } catch (FileNotFoundException ex) {
            System.out.println("[ERROR]initialiseArcade cannot find file '"+
                arcadeGamesFile+"' and or '" +customersFile+"'.");
            System.exit(-1);
        }


        try {
            simulateFun(arcade, transactionsFile);
        } catch (FileNotFoundException ex) {
            System.out.println("[ERROR]simulateFun cannot find file '"+
                transactionsFile+"'.");
            System.exit(-1);
        }

        System.out.println("\n\n==========================================================");
        Arcade.printCorporateJargon();

        int[] stats = arcade.countArcadeGames();
        System.out.println("total number of cabinetgames in this arcade: " + stats
            [0]);
        System.out.println("number of active games in this arcade (not including vr):
            " + stats[1]);
        System.out.println("number of virtual reality games in this arcade:" + stats
            [2]);

        System.out.println("the richest customer is: " + arcade.findRichestCustomer()
            );

        //using printf as to always keep 2 decimal places
        System.out.printf("the median price is: £%.2f\n", ((double)arcade.
            getMedianGamePrice()) / 100);

        System.out.println("the total revenue is: £" + ((double) arcade.getRevenue())
             / 100);
        System.out.println("==========================================================\n\n");

    }

    public static ArrayList<String> readFromFile(File fileObj) throws
        FileNotFoundException{
        ArrayList<String> contents = new ArrayList<>();
        // code copied and then modified from https://www.w3schools.com/java/
            java_files_read.asp

        try(Scanner myReader = new Scanner(fileObj)){
            while (myReader.hasNextLine()) {
                String current_line = myReader.nextLine();
                contents.add(current_line);  // added this line
            }
            myReader.close();
        }

        return contents;
    }

    public static Arcade initialiseArcade(String arcadeName, File gamesFile, File
        customerFile) throws FileNotFoundException{
        Arcade newArcadeObj = new Arcade(arcadeName);

        ArrayList<String> gamesFileContens = readFromFile(gamesFile);
        // for dealing with a
        for (String line : gamesFileContens) {
            ArcadeGame arcadeGameToAdd = null;
            // each line goes Id, name,typeofgame, cost, age restriction
            // if vr then there is a extra tracking type
            // if cabinet then there is no age restriction but a yes or no for giving
                out rewards.
            String[] lineData = line.split("@");
            String gameId = lineData[0];
            String gameName = lineData[1];
            String gameType = lineData[2];
            int pricePerPlay = Integer.parseInt(lineData[3]);
            int ageRequirement;

            // creating the virtualRealityGame/CabinetGame/ActiveGame objects to the
```

```
                expected ArcadeGame object, arcadeGameToAdd.
            switch (gameType) {
                case "virtualReality" -> {
                    ageRequirement = Integer.parseInt(lineData[4]);
                    String trackingType = lineData[5];
                    try {
                        arcadeGameToAdd = new VirtualRealityGame(gameId, pricePerPlay
                            , gameName, ageRequirement, trackingType);
                    } catch (InvalidGameIdException e) {
                        System.out.println("[Error]VirtualRealityGame constructor" +
                            e);
                        continue;
                    }
                }

                case "cabinet" -> {
                    boolean givesReward = lineData[4].equals("yes");
                    try {
                        arcadeGameToAdd = new CabinetGame(gameId, pricePerPlay,
                            gameName, givesReward);
                    } catch (InvalidGameIdException e) {
                        System.out.println("[Error]CabinetGame constructor" + e);
                        continue;
                    }
                }

                case "active" -> {
                    ageRequirement = Integer.parseInt(lineData[4]);
                    try {
                        arcadeGameToAdd = new ActiveGame(gameId, pricePerPlay, gameId
                            , ageRequirement);
                    } catch (InvalidGameIdException e) {
                        System.out.println("[Error]ActiveGame constructor" + e);
                        continue;
                    }
                }
            }

            if (arcadeGameToAdd != null){
                // will stay null if it couldnt initilise
                System.out.println("adding arcadegame: " + arcadeGameToAdd);
            }

            newArcadeObj.addArcadeGame(arcadeGameToAdd);
        }


        ArrayList<String> customerFileContens = readFromFile(customerFile);
        for (String elem : customerFileContens) {
            String[] lineData = elem.split("#");
            String accountId = lineData[0];
            String name = lineData[1];
            int initalBalance = Integer.parseInt(lineData[2]);
            int age = Integer.parseInt(lineData[3]);
            String discountType = "NONE";
            Customer customerToAdd;
            if (lineData.length == 5){
                discountType = lineData[4];
                customerToAdd = new Customer(accountId, name, age, discountType,
                    initalBalance);
            }
            else{
                customerToAdd = new Customer(accountId, name, age, discountType,
```

```java
                    initalBalance);
            }
            newArcadeObj.addCustomer(customerToAdd);
        }

        return newArcadeObj;
    }

    public static void simulateFun(Arcade arcade, File transactionFile) throws
       FileNotFoundException{
        ArrayList<String> transactionFileContense = readFromFile(transactionFile);

        for (String line : transactionFileContense) {
            String[] lineData = line.split(",");
            String command = lineData[0];
            String customerId = lineData[1];
            switch (command) {
                case "PLAY" -> {
                    String gameId = lineData[2];
                    boolean peakTime = lineData[3].equals("PEAK");

                    arcade.processTransaction(customerId, gameId, peakTime);
                }
                case "NEW_CUSTOMER" -> {
                    String name = lineData[2];
                    int age;
                    String discountType;
                    int initalBalance;

                    if (lineData.length == 5){
                        discountType = "NONE";
                        initalBalance = Integer.parseInt(lineData[3]);
                        age = Integer.parseInt(lineData[4]);
                    }
                    else{
                        discountType = lineData[3];
                        initalBalance = Integer.parseInt(lineData[4]);
                        age = Integer.parseInt(lineData[5]);
                    }

                    Customer newCustomer = new Customer(customerId, name, age,
                       discountType, initalBalance);

                    arcade.addCustomer(newCustomer);
                }
                case "ADD_FUNDS" -> {
                    int moneyToAdd = Integer.parseInt(lineData[2]);

                    try {
                        arcade.getCustomer(customerId).AddFunds(moneyToAdd);
                    } catch (InvalidCustomerException ex) {
                        System.out.println("[Error]from getCustomer: " + ex);

                        System.out.println("could not add £"+(double)moneyToAdd /100+
                           " as we could not find that Id.");
                    }
                }

            }
        }
    }

}
```

ArcadeGame.java

```java
/*======================================================

File                    :  ArcadeGame.java

date                    :  14/4/2025

Author                  :  Benedict Ward

Description             :  this class counts up to 10 marks
                           the simple constuctor accessor and mutator methods.
                           with the class it's self being abstract along with
                              calculatePrice
                           for "basis for the cabinet game and active game
                              subclasses".

Possible Exceptions     :

History                 :  28/2/2025 v1.0 - added the constructor and calculatePrice
                                            plus accessor and mutator methods.
                                            8:53pm added the this keyword in the
                                               accessor methods

                           1/3/2025 v1.01 - moved isAllAlphanumeric here as a
                              protected function
                                            as all subclasses have the function

                           13/3/2025 v1.02 - gameId,pricePerPlay,name attributes are
                               all final
                                             removed the setter methods for those
                                                attributes

                           19/3/2025 v1.03 - all attributes are now privated not
                              protected
                                             and getter methods are all protected
======================================================*/



public abstract class ArcadeGame{
    private final String gameId;
    private final int pricePerPlay;
    private final String name;
    public ArcadeGame(String gameId, int pricePerPlay, String name){
        this.gameId = gameId;
        this.pricePerPlay = pricePerPlay;
        this.name = name;
    }

    protected abstract int calculatePrice(boolean peak);

    protected boolean isAllAlphanumeric(String str){
        // gets each character of a given String and checks if its a digit or a
           letter this stops unique characters
        for (int i = 0; i < str.length(); i++) {
            if (!(Character.isDigit(str.charAt(i)) || Character.isLetter(str.charAt(i
               )))){
                return false;
            }
        }
        return true;
    }

    protected String getGameId() {
        return this.gameId;
    }

    protected int getPricePerPlay() {
        return this.pricePerPlay;
    }

    protected String getName() {
        return this.name;
    }
}
```

CabinetGame.java

```java
/*==================================================

File                        :  CabinetGame.java

date                        :  14/4/2025

Author                      :  Benedict Ward

Description                 :  this class and ActiveGame counts up to 10 marks,

Possible Exceptions         :  InvalidGameIdException from CabinetGame

History                     :  28/2/2025 v1.0 - added code
                                                4:17pm fixed edge case where the
                                                    characters
                                                where not checked only the length was
                                                9:11pm fixed the isAllAphanumeic
                                                    function and logic using that value
                                                10:58pm fixed the toString method by
                                                    removing the format function.

                               3/1/2025 v1.01 - moved the helper function
                                   isAllAlphanumeric
                                                to ArcadeGame where it get inherritted
                                                    from
                                                fixed calculatePrice, missing ! for
                                                    boolean logic and wrongful cast to
                                                    int not double.

                               21/3/2025 v1.02 - added final keyword to the class

                               11/04/2025 v1.03 - better toString
==================================================*/



public final class CabinetGame extends ArcadeGame{
    private final boolean givesReward;  // only needs a accessor method
    public CabinetGame(String gameId, int pricePerPlay, String Name, boolean
        givesReward) throws InvalidGameIdException{
        super(gameId,pricePerPlay,Name);

        this.givesReward = givesReward;

        //validating gameId
        if(!gameId.startsWith("C")){
            throw new InvalidGameIdException("gameId invalid, does not start is a 'C
                '.");
        }
        else if(!(isAllAlphanumeric(gameId) && (gameId.length() == 10))){
            throw new InvalidGameIdException("gameId invalid, does not contain
                exactly 10 alphanumeric characters.");
        }
    }

    @Override
    protected int calculatePrice(boolean isPeakHour) {
        boolean canGetDiscounted = !isPeakHour;
        double totalDiscount = 1;
        if (canGetDiscounted){

            // 20% discount if the game gives out rewards
            // else 50%
            if (getGivesReward()){
                totalDiscount -= 0.20;
            }
            else{
                totalDiscount -= 0.50;
            }
        }
        // to round down
        return (int) Math.floor(getPricePerPlay() * totalDiscount);
    };

    public boolean getGivesReward() {
        return this.givesReward;
    }

    @Override
    public String toString(){
        return this.getClass().getSimpleName()+"{gameId: "+this.getGameId()+",
           pricePerPlay: "+this.getPricePerPlay()+", Name: "+this.getName()+",
           GiveReward: "+this.getGivesReward()+"}";

    }

    public static void main(String[] args){
        // expected result: pass, as this is all typical data
        CabinetGame gameIdTest1;
        try{
            gameIdTest1 = new CabinetGame("CBGCR27FQM",200,"GAMENAME", true);
            System.out.println(gameIdTest1.toString());
        }catch(InvalidGameIdException e){
            System.out.println(e);
        }
        // actual result: i was correct, toString executed without error.


        // expected result: fail as gameId does not start with C
        CabinetGame gameIdTest2;
        try{
            gameIdTest2 = new CabinetGame("BBGCR27FQM",200,"GAMENAME", true);
            System.out.println(gameIdTest2.toString());
        }catch(InvalidGameIdException e){
            System.out.println(e);
        }
        // actual result: i was correct, "gameId invalid, does not start is a 'C'."

        // epected result: fail as gameId is too long.
        CabinetGame gameIdTest3;
        try{
            gameIdTest3 = new CabinetGame("CBGCR27FQMMMM",200,"GAMENAME", true);
            System.out.println(gameIdTest3.toString());
        }catch(InvalidGameIdException e){
            System.out.println(e);
        }
        // actual result: i was correct, "gameId invalid, does not contain exactly 10
             alphanumeric characters."


        CabinetGame calculatePriceTest1;
        CabinetGame calculatePriceTest2;
        try{
            calculatePriceTest1 = new CabinetGame("CBGCR27FQM",200,"GAMENAME", true);
            calculatePriceTest2 = new CabinetGame("CBGCR27FQM",200,"GAMENAME", false)
               ;
            boolean isPeakHour = true;
            System.out.println("expected price of  200, actual price of " +
               calculatePriceTest1.calculatePrice(isPeakHour));  // 200
            System.out.println("expected price of  160, actual price of " +
               calculatePriceTest1.calculatePrice(!isPeakHour));  // 160

            System.out.println("expected price of  200, actual price of " +
               calculatePriceTest2.calculatePrice(isPeakHour));  // 200
            System.out.println("expected price of  100, actual price of " +
               calculatePriceTest2.calculatePrice(!isPeakHour));  //100
        }catch(InvalidGameIdException e){
            System.out.println(e);
        }
        // actual output was, 0,200 and 0,200
        // fix: missing ! when setting canGetDiscounted and was casting totalDiscount
            to int not double
        // causing any discount to set totalDiscount to 0
        // after re-running i get the output 200,160 and 200,100 as expected.
    }

}
```

ActiveGame.java

```java
/*==================================================

File                    :  ActiveGame.java

date                    :  28/2/2025

Author                  :  Benedict Ward

Description             :  this class and CabinetGame counts up to 10 marks


Possible Exceptions     :  InvalidGameIdException from ActiveGame()


History                 :  28/2/2025 v1.0 - added code
                                         4:19pm fixed edge case where the
                                            characters
                                         where not checked only the length was.

                           28/2/2025 v1.01 - added possible exceptions in the header
                                          added a toString method
                                          9:08 fixed the isAllAphanumeic function
                                              and logic using that value.

                           1/3/2025 v1.02 - added final keyword to ageRequirement
                                         finished testing in the main function.

                           1/3/2025 v1.02 - moved isAllalphanumeric to ArcadeGame
                              where
                                         it will be inherited from.
                                         calculatePrice now uses getPricePerPlay
                                            () not this.pricePerPlay.

                           13/3/2025 v1.03 - more checking in the constructor method
                                          for the gameId to not start with 'AV'
                                             as that is
                                          a different class. then removed this as
                                              virtualRealityGame
                                          extended from this class.

                           19/3/2025 v1.04 - made ageRequirement private

                           11/04/2025 v1.05 - better toString
==================================================*/


public class ActiveGame extends ArcadeGame{
    private final int ageRequirement;  // only needs accessor for this field not
        setter
    public ActiveGame(String gameId, int pricePerPlay, String name, int
        ageRequirement) throws InvalidGameIdException{
        super(gameId,pricePerPlay,name);
        this.ageRequirement = ageRequirement;

        if(!gameId.startsWith("A")){
            throw new InvalidGameIdException("gameId invalid, does not start is a 'A
                '.");
        }
        else if(!(isAllAlphanumeric(gameId) && (gameId.length() == 10))){
            throw new InvalidGameIdException("gameId invalid, does not contain
                exactly 10 alphanumeric characters.");
        }
    }

    protected int getAgeRequirement() {
        return this.ageRequirement;
    }

    @Override
    protected int calculatePrice(boolean isPeakHour) {
        boolean canGetDiscounted = !isPeakHour;

        if(canGetDiscounted){
            return (int) (getPricePerPlay() * 0.8);  // 20% discount
        }

        return getPricePerPlay();
    }

    @Override
    public String toString(){
        return this.getClass().getSimpleName()+"{gameId: "+this.getGameId()+",
            pricePerPlay: "+this.getPricePerPlay()+", Name: "+this.getName()+",
            ageRequirement: "+this.getAgeRequirement()+"}";
    }

    public static void main(String[] args) {

        // testing
        // expected result: pass, as it is given the data from the file
        try {
           ActiveGame gameIdTest1 = new ActiveGame("AHW0HK1F03",80,"Foosball",3);
           System.out.println(gameIdTest1.getAgeRequirement());
        } catch (InvalidGameIdException ex) {
           System.out.println("invalid gameid");
        }
        // actual result: i was correct, no error was raised

        // expected result: InvalidGameIdException will get raised
        try {
           ActiveGame gameIdTest2 = new ActiveGame("BHW0HK1F03",80,"Foosball",3);
           gameIdTest2.getName();
        } catch (InvalidGameIdException ex) {
           System.out.println("error" + ex);
        }
        // actual result: i was correct, an error was raised as gameId started with a
           B

        // expected result: InvalidGameIdException will get raised due to length.
        try {
           ActiveGame gameIdTest3 = new ActiveGame("AHW0HK1F033",80,"Foosball",3);
           gameIdTest3.getName();
        } catch (InvalidGameIdException ex) {
           System.out.println("error"+ex);
        }
        // actual result: i was correct an error was raised "gameId invalid, does not
           contain exactly 10 alphanumeric characters."


        // testing for calculatePrice with a valid ActiveGame
        ActiveGame validgame = null;
        try {
```

```
            validgame = new ActiveGame("AHW0HK1F03",80,"Foosball",3);
        } catch (InvalidGameIdException ex) {
            System.out.println("invalid gameid");
        }

        boolean isPeakHour=true;
        System.out.println("expected price 80, actual price :"+validgame.
            calculatePrice(isPeakHour));  // 80
        System.out.println("expected price 64, actual price :"+validgame.
            calculatePrice(!isPeakHour));  // 64
        // actual result: i was correct calculatePrice gave the expected price
    }
}
```

VirtualRealityGame.java

```java
/*====================================================

File              :  VirtualRealityGame.java

date              :  28/2/2025

Author            :  Benedict Ward

Description       :  worth upto 5 marks, this class will handle all VR games the only
    unique
                     thing about this class is that ControlType is a enum of
                         EnumControlTypes

History           :  28/2/2025 v1.0 - added all the code then did the testing as shown
     in the
                                       main when commented out
                                       6:05pm fixed the tostring saying cabinetgame obj
                                           .
                                       8:036pm fixed the constructor not asking for
                                           ageRequirement
                                       and just parsing in pricePerPlay twise
                                       9:11 fixed the isAllAphanumeic function and
                                           logic using that value
                                       10:58pm fixed the toString method by removing
                                           the format function.

                     1/3/2025 v1.01 - removed isAllAlphanumeric as it gets inherited
                         from ArcadeGame

                     21/3/2025 v1.02 - added final keyword to the class

                     11/04/2025 v1.03 - better toString
===================================================*/



public final class VirtualRealityGame extends ActiveGame{

    private EnumControlTypes ControlType;  // cant be final as the switch case cannot
         have a defualt statement
    private enum EnumControlTypes {  HEADSETONLY,
                                     FULLBODYTRACKING,
                                     HEADSETANDCONTROLLER};


    public VirtualRealityGame(String gameId, int pricePerPlay, String Name, int
        ageRequirement, String ControlType) throws InvalidGameIdException{
        super(gameId, pricePerPlay, Name, ageRequirement);

        switch (ControlType) {
            case "headsetOnly" -> this.ControlType = EnumControlTypes.HEADSETONLY;
            case "fullBodyTracking" -> this.ControlType = EnumControlTypes.
                FULLBODYTRACKING;
            case "headsetAndController" -> this.ControlType = EnumControlTypes.
                HEADSETANDCONTROLLER;
        }

        //validating gameId
        if(!gameId.startsWith("AV")){
            throw new InvalidGameIdException("gameId invalid, does not start is a 'AV
                '.");
        }
        else if(!(isAllAlphanumeric(gameId) && (gameId.length() == 10))){
            throw new InvalidGameIdException("gameId invalid, does not contain
                exactly 10 alphanumeric characters.");
        }
    }

    public boolean isHeadsetOnly(){
        return this.ControlType == EnumControlTypes.HEADSETONLY;
    }

    public boolean isFullBodyTracking(){
        return this.ControlType == EnumControlTypes.FULLBODYTRACKING;
    }

    public boolean isHeadsetAndController(){
        return this.ControlType == EnumControlTypes.HEADSETANDCONTROLLER;
    }

    private EnumControlTypes getControlType(){
        return this.ControlType;
    }

    @Override
    protected int calculatePrice(boolean isPeakHour) {
        boolean canGetDiscounted = !isPeakHour;
        double totalDiscount = 1;
        if (canGetDiscounted){
            if(isHeadsetOnly()){
                totalDiscount -= 0.10;
            }
            else if(isHeadsetAndController()){
                totalDiscount -= 0.05;
            }
        }
        return (int) Math.floor(getPricePerPlay() * totalDiscount);
    }

    @Override
    public String toString(){
        return this.getClass().getSimpleName()+"{gameId: "+this.getGameId()+",
           pricePerPlay: "+this.getPricePerPlay()+", Name: "+this.getName()+",
           ControlType: "+getControlType()+"}";
    }

    public static void main(String[] args){
        //Testing took place on 28/02/25 around 12-1:30

        // expected restult: error, InvalidgameId as gameId does not start with a AV
           everything else should be valid though
        try {
            VirtualRealityGame gameIdTest1 = new VirtualRealityGame("gameId",200,"
                GAMENAME",0,"headsetOnly");
            System.out.println(gameIdTest1);
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }
        // given result: i was correct, "InvalidGameIdException: gameId invalid, does
            not start is a 'C'."

        // expected restult: error InvalidgameId as gameId does not contain 10
           alphanumeric characters.
        try {
            VirtualRealityGame gameIdTest2 = new VirtualRealityGame("CgameId",200,"
                GAMENAME",0,"headsetOnly");
            System.out.println(gameIdTest2.getClass());
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }
        // given result: i was incorrect, "InvalidGameIdException: gameId invalid,
            does not start is a 'C'.".
        // fix: simple spelling mistake and i will now input the correct gameId,

        // expected result: will throw an error for invalid String length.
        try {
            VirtualRealityGame gameIdTest3 = new VirtualRealityGame("AVgameId",200,"
                GAMENAME", 0,"headsetOnly");
            System.out.println(gameIdTest3.getClass());
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }


        // given result: i was correct, "InvalidGameIdException: gameId invalid, does
             not contain exactly 10 alphanumeric characters."

        // expected restult: incorrectly passes, as i am incorrectly checking for
            alphanumeric characters by just checking the length.
        try {
            VirtualRealityGame gameIdTest4 = new VirtualRealityGame(" AVgameId",200,"
                GAMENAME", 0,"headsetOnly");
            System.out.println(gameIdTest4.getClass());
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }

        // given restuls: i was correct, no error message means it passes when it
            shouldnt of.
        // fix: i will rework/make the function to check the alphanumeric characters
            instead of just using .length()


        // expected result: pass as this is all valid
        VirtualRealityGame ControlTypeTest1;
        try {
            ControlTypeTest1 = new VirtualRealityGame("AVI1USPBNG", 0, "Virtual UEA
                Tour", 0, "headsetOnly");
            System.out.println(ControlTypeTest1.getControlType());
            System.out.println(ControlTypeTest1);
        } catch (InvalidGameIdException ex) {
            System.out.println(ex);
        }
        // given result: i was correct, output is HEADSETONLY

        // expected result: fail as headsetOnly is incorrectly capitalised so no
            value is set
        VirtualRealityGame ControlTypeTest2;
        try {
            ControlTypeTest2 = new VirtualRealityGame("AVI1USPBNG", 0, "Virtual UEA
                Tour", 0, "hEaDsEtOnly");
            System.out.println(ControlTypeTest2.getControlType());
        } catch (InvalidGameIdException ex) {
            System.out.println(ex);
        }
        // given result: i was correct, output is null
```

```java

        //testing calculatePrice when given a valid VirtualRealityGame object
        VirtualRealityGame calculatePriceTest1;
        VirtualRealityGame calculatePriceTest2;
        VirtualRealityGame calculatePriceTest3;
        try {
            calculatePriceTest1 = new VirtualRealityGame("AVI1USPBNG", 100, "Virtual
                UEA Tour",0, "headsetOnly");
            calculatePriceTest2 = new VirtualRealityGame("AVI1USPBNG", 100, "Virtual
                UEA Tour",0, "fullBodyTracking");
            calculatePriceTest3 = new VirtualRealityGame("AVI1USPBNG", 100, "Virtual
                UEA Tour",0, "headsetAndController");
            System.out.println(calculatePriceTest1.getClass());
            System.out.println(calculatePriceTest2.getClass());
            System.out.println(calculatePriceTest3.getClass());
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }

        // boolean isPeakHour = true;
        // System.out.println("expected price of  100, actual price of " +
            calculatePriceTest1.calculatePrice(isPeakHour));  //100
        // System.out.println("expected price of  90, actual price of " +
            calculatePriceTest1.calculatePrice(!isPeakHour));  // 90

        // System.out.println("expected price of  100, actual price of " +
            calculatePriceTest2.calculatePrice(isPeakHour));  // 100
        // System.out.println("expected price of  100, actual price of " +
            calculatePriceTest2.calculatePrice(!isPeakHour));  // 100

        // System.out.println("expected price of  100, actual price of " +
            calculatePriceTest3.calculatePrice(isPeakHour));  // 100
        // System.out.println("expected price of  95, actual price of " +
            calculatePriceTest3.calculatePrice(!isPeakHour));  // 95
        //fix: missing ! when setting canGetDiscounted + wrongful cast to int for
            totalDiscount. now casts to double
        // after these corrections i get the correct output of 100,90 and 100,100 and
             100,95



        //error stats:
        // gameIdTest       expected pass rate : actual pass rate    (75%)
        //                                     4 : 3

        // ControlTypeTest  expected pass rate : actual pass rate    (100%)
        //                                     2 : 2

        // where expected pass rate means i expect one result
        // and atual pass is when the result is the expected
    }
}
```

Customer.java

```
/*==================================================

File                    :  Customer.java

date                    :  14/4/2025

Author                  :  Benedict Ward

Description             :  worth up to 20 marks

Possible Exceptions     :  InsufficientBalanceException from chargeAccount
                           AgeLimitException from chargeAccount


History                 :  28/2/2025 v1.0 - finished coding at 3:30pm now doing
   testing
                                          3:50 found a
                                          10:59pm fixed the toString method by
                                             removing the format function.

                           1/3/2025 v1.0 - chargeAccount now uses getAccountBalance
                              () instead
                                          of this.accountBalance, same with age.

                           13/3/2025 v1.1 - now using .getClass().getSimpleName() to
                              get
                                          the class name instead of checking the
                                             class's
                                          toString() result

                           21/3/2025 v1.11 - added final keyword to the class

                           11/04/2025 v1.12 - better toString
==================================================*/



public final class Customer {
    private final String accountId;
    private final String Name;
    private final int Age;

    private final EnumPersonalDiscounts personalDiscount;
    private enum EnumPersonalDiscounts {NONE,STAFF,STUDENT}  //REMINDER students will
        be allowed a negative balance of upto -500
    private int accountBalance;  // 100 = £1

    public Customer(String accountId, String Name, int Age, String discountType){
        this.accountId = accountId;
        this.Name = Name;
        this.Age = Age;
        this.accountBalance = 0;

        //checking what discountType was given
        switch (discountType) {
            case "STUDENT" -> this.personalDiscount = EnumPersonalDiscounts.STUDENT;
            case "STAFF" -> this.personalDiscount = EnumPersonalDiscounts.STAFF;
            default -> this.personalDiscount = EnumPersonalDiscounts.NONE;
        }
    }
    public Customer(String accountId, String Name, int Age, String discountType, int
        initalBalance){
        this.accountId = accountId;
        this.Name = Name;
        this.Age = Age;
        // this math.max function is used so the value can not be smaller then 0 but
            can go anywhere higher.
        this.accountBalance = Math.max(0,initalBalance);

        //checking what discountType was given
        switch (discountType) {
            case "STUDENT" -> this.personalDiscount = EnumPersonalDiscounts.STUDENT;
            case "STAFF" -> this.personalDiscount = EnumPersonalDiscounts.STAFF;
            default -> this.personalDiscount = EnumPersonalDiscounts.NONE;
        }
    }

    public void AddFunds(int amount){
        //checking if positive
        if (0 < amount){
            this.accountBalance += amount;
        }
    }

    public int chargeAccount(ArcadeGame arcadeGameObj, boolean peakTime) throws
        InsufficientBalanceException, AgeLimitException{
        double discountFactor = 1;
        boolean canGoNegative = false;

        // staff get 10% off, students get 5% off.
        if (isDiscountStaff()){
            discountFactor -= 0.10;
        }
        if (isDiscountStudent()){
            discountFactor -= 0.05;
            canGoNegative = true;
        }

        int fullPrice = arcadeGameObj.calculatePrice(peakTime);

        int price = (int) (Math.floor(fullPrice * discountFactor));

        if (0 < (getAccountBalance() - price) || (-500 < (getAccountBalance() - price
            ) && canGoNegative)) {
            // the user has enough funds to pay

            // now checking if the arcadegameObj is either activegame,cabinetgame or
                virtualrealitygame then type casting it to a new variable
            //.getClass().getSimpleName() returns the class name
            if (arcadeGameObj.getClass().getSimpleName().equals("ActiveGame")){
                ActiveGame activeGameObj = (ActiveGame) arcadeGameObj;
                int ageRequirement = activeGameObj.getAgeRequirement();

                if (ageRequirement > getAge()){
                    throw new AgeLimitException("you must be at least " +
                        ageRequirement + ", to play this game, you are only " + this.
                        Age);
                }
            }

            this.accountBalance -= price;
            return price;
        }
        else{
            throw new InsufficientBalanceException("the price is," + price + ". and
                you only have, " + getAccountBalance());
        }
    }

    public boolean isDiscountNone(){
        return this.personalDiscount == EnumPersonalDiscounts.NONE;
    }

    public boolean isDiscountStaff(){
        return this.personalDiscount == EnumPersonalDiscounts.STAFF;
    }

    public boolean isDiscountStudent(){
        return this.personalDiscount == EnumPersonalDiscounts.STUDENT;
    }

    public String getAccountId(){
        return this.accountId;
    }
    public String getName(){
        return this.Name;
    }
    public int getAge(){
        return this.Age;
    }
    public int getAccountBalance(){
        return this.accountBalance;
    }
    private EnumPersonalDiscounts getPersonalDiscount(){
        return this.personalDiscount;
    }

    @Override
    public String toString(){
        return this.getClass().getSimpleName()+"{accountID: "+this.getAccountId()+",
            name: "+this.getName()+", age: "+this.getAge()+", discounttype: "+this.
            getPersonalDiscount()+", balance "+this.getAccountBalance()+"}";

    }

    public static void main(String[] args){

        // this is a test for when given a valid arcadegame does charging the
            customer work correctly
        // expected result: it will loop 2 times like normal, on the 3rd it will
            throw a InsufficientBalanceException


        ArcadeGame ag = null;
        try {
            ag = new ActiveGame("AL2ETWHG0Q", 200, "Name",18);
        } catch (InvalidGameIdException ex) {
        }
        Customer customer = new Customer("accountID", "Name", 18,"NONE",500);

        for (int i = 0; i < 4; i++) {
            try {
                customer.chargeAccount(ag, true);
                System.out.println(i +" : "+ customer.toString());
            } catch (AgeLimitException | InsufficientBalanceException e) {
```

```java
                System.out.println("[Error]when charging account");
            }
        }
        // actual result: looped 3 times then gave an error because the pricePerPlay
            was discounted and i forgot to account for that
        // correction: well it shouldnt of been discounted as its peakTime,
        // fix: added a not to canGetDiscounted in ActiveGame
        // after re running it gave the expected result of looping 2 times, error on
            the 3rd.

        // same testing but the Customers discount type is now Student
        // expected result: it will manage to loop all 4 times with the balance going
             negative


        ArcadeGame ag2 = null;
        try {
            ag2 = new ActiveGame("AL2ETWHG0Q", 200, "Name",18);
        } catch (InvalidGameIdException ex) {
        }
        Customer customer2 = new Customer("accountID", "Name", 18,"STUDENT",500);

        for (int i = 0; i < 4; i++) {
            try {
                customer2.chargeAccount(ag2, true);
            } catch (InsufficientBalanceException | AgeLimitException e) {
                System.out.println("[ERROR]"+e);
            }
           System.out.println(i +" : "+ customer2.toString());
        }
        // given result: i was correct, looped 4 times with balance going from
            500->310->120->-70->-260
        // other notes: swapped out the 4 for a 7 and i only reached to 4 before the
            balance hit -450
        // meaning it couldnt go lower and error was thrown "
            InsufficientBalanceException: the price is,190. and you only have, -450"
    }

}
```

Arcade.java

```java
/*=====================================================

File                        :  Arcade.java

date                        :  14/4/2025

Author                      :  Benedict Ward

Description                 :  worth upto 25 marks, mainly linking up the types of
    arcade games
                               to customers

Possible Exceptions         :  InvalidCustomerException from getCustomer
                               InvalidGameIdException from getArcadeGame


History                     :  28/2/2025 v1.0 - 4:04 started, added the custom exception
    ,
                                                made the constructors + getCustomer
                                                4:33 started testing getCustomer
                                                5:51 back on the grind :3
                                                11:34pm adding functionality for getting
                                                    the median
                               1/3/2025 v1.01 - created private acessor method
                                   getMedianGamePrice

                               13/3/2025 v1.02 - fixed getMedianPrice()
                                                 the error was casting to a double after
                                                     dividing the int
                                                 when it should be casting to a double
                                                    then dividing by 2.

                                                 fixed countArcadeGames()
                                                 now using .getClass().getSimpleName().
                                                    equals instead of checking
                                                 the toString output

                               14/3/2025 v1.03 - added a toString method()

                               21/3/2025 v1.02 - added final keyword to the class

                               4/5/2025 v.1.1 - added hashmap for ArcadeGameCollection
                                   and customerCollection
=====================================================*/
import java.util.Arrays;
import java.util.HashMap;


public final class Arcade {
    private final String arcadeName;
    private final HashMap<String, ArcadeGame> ArcadeGameCollection;
    private final HashMap<String, Customer> customerCollection;
    private int revenue;  // cant be final as revenue will change

    public Arcade(String arcadeName){
        this.arcadeName = arcadeName;
        this.customerCollection = new HashMap<>();
        this.ArcadeGameCollection = new HashMap<>();
        this.revenue = 0;
    }

    public void addCustomer(Customer customer){
        this.customerCollection.put(customer.getName(),customer);
    }

    public void addArcadeGame(ArcadeGame arcadeGame){
        this.ArcadeGameCollection.put(arcadeGame.getGameId(),arcadeGame);
    }

    public Customer getCustomer(String customerID) throws InvalidCustomerException{
        for (Customer elem : getCustomerCollection().values()) {
            if (elem.getAccountId().equals(customerID)) {
                return elem;
            }
        }
        throw new InvalidCustomerException("No customer found with the ID of " +
           customerID);
    }

    public ArcadeGame getArcadeGame(String gameId) throws InvalidGameIdException{
        if (this.ArcadeGameCollection.get(gameId) == null){
            throw new InvalidGameIdException("No game found with the ID of " + gameId
               );
        }
        return this.ArcadeGameCollection.get(gameId);
    }

    public Customer findRichestCustomer(){
        int highestBalance = -501;  // not setting it to 0 as Students can have -500
        Customer richestCustomer = null;
        for (Customer customer : getCustomerCollection().values()) {
            if (customer.getAccountBalance() > highestBalance){
                highestBalance = customer.getAccountBalance();
                richestCustomer = customer;
            }
        }
        return richestCustomer;
    }

    private HashMap<String,ArcadeGame> getArcadeGameCollection(){
        return this.ArcadeGameCollection;
    }

    public int getMedianGamePrice(){
        int[] allPrices = new int[getArcadeGameCollection().size()];
        int index = 0;
        for (ArcadeGame arcadeGameKey : getArcadeGameCollection().values()) {
            allPrices[index] = (arcadeGameKey.getPricePerPlay());
            index += 1;
        }

        Arrays.sort(allPrices);

        if (((double) (getArcadeGameCollection().size())) / 2 ==
           getArcadeGameCollection().size() / 2) {
            // when there is an even amount of ArcadeGame machines
            return (allPrices[getArcadeGameCollection().size() / 2] + allPrices[(
               getArcadeGameCollection().size() + 1) / 2]) / 2;
        }
        else{
            // when there is an odd amount of ArcadeGame machines
            return allPrices[getArcadeGameCollection().size() / 2];

        }
    }

    public int[] countArcadeGames(){
        int totalCabinetGames = 0;
        int totalActiveGames = 0;
        int totalVirtualgames = 0;

        for (ArcadeGame arcadegame : getArcadeGameCollection().values()) {
            if (arcadegame.getClass().getSimpleName().equals("ActiveGame")){
                totalCabinetGames += 1;
            }

            if (arcadegame.getClass().getSimpleName().equals("ActiveGame")){
                totalActiveGames += 1;
            }

            if (arcadegame.getClass().getSimpleName().equals("VirtualRealityGame")){
                totalVirtualgames += 1;
            }
        }

        int[] toReturn = {totalCabinetGames, totalActiveGames ,totalVirtualgames};
        return toReturn;
    }

    public static void printCorporateJargon(){
        System.out.println("GamesCo does not take responsibility for any accidents or
             fits of rage that occur on the premises");
    }

    public String getArcadeName() {
        return this.arcadeName;
    }

    public int getRevenue() {
        return this.revenue;
    }

    public boolean processTransaction(String customerId, String gameId, boolean peak)
       {
        ArcadeGame arcadeGameObj;
        Customer customer;
        int amountCharged;
        try {
            arcadeGameObj = getArcadeGame(gameId);
        } catch (InvalidGameIdException e) {
            return false;
        }

        try {
            customer = getCustomer(customerId);
        } catch (InvalidCustomerException e) {
            return false;
        }

        try {
            amountCharged = customer.chargeAccount(arcadeGameObj, peak);
        } catch (InsufficientBalanceException | AgeLimitException e) {
            return false;
        }

        this.revenue += amountCharged;
```

```java
        return true;
    }

    public HashMap<String,Customer> getCustomerCollection(){
        return this.customerCollection;
    }

    @Override
    public String toString(){
        return "this is a Arcade object, arcadeName " + getArcadeName() + "
            ArcadeGameCollection size: " + getArcadeGameCollection().size() + ",
            customerCollection size: " + getCustomerCollection().size() + ", revenue:
             "+ getRevenue();
    }
    public static void main(String[] args){
        // a test for the addCustomer along with getCustomer
        Customer customer1 = new Customer("748A66", "name1", 18, "STUDENT",500);
        Customer customer2 = new Customer("1C6498", "name2", 18, "STUDENT",500);
        Customer customer3 = new Customer("305459", "name3", 18, "STUDENT",500);
        Customer customer4 = new Customer("203685", "name4", 18, "STUDENT",500);
        Arcade arcade = new Arcade("arcadeName");
        arcade.addCustomer(customer1);
        arcade.addCustomer(customer2);
        arcade.addCustomer(customer3);
        arcade.addCustomer(customer4);
        try {
            System.out.println(arcade.getCustomer("203685"));
            System.out.println(arcade.getCustomer("000000"));  // this line correctly
                 throws an error
        } catch (InvalidCustomerException e) {
            System.out.println("caught an error: "+e);
        }

        try {
            ArcadeGame activeGame1 = new ActiveGame("AHW0HK1F01",100,"Foosball",3);
            ArcadeGame activeGame2 = new ActiveGame("AHW0HK1F02",90,"Foosball",3);
            ArcadeGame activeGame3 = new ActiveGame("AHW0HK1F03",80,"Foosball",3);
            ArcadeGame activeGame4 = new ActiveGame("AHW0HK1F04",70,"Foosball",3);



            arcade.addArcadeGame(activeGame1);
            arcade.addArcadeGame(activeGame2);
            arcade.addArcadeGame(activeGame3);


            System.out.println(arcade.getArcadeGame("AHW0HK1F01"));

            System.out.println("median:" + arcade.getMedianGamePrice());

            arcade.addArcadeGame(activeGame4);

            System.out.println("median:" + arcade.getMedianGamePrice());
        } catch (InvalidGameIdException e) {
            System.out.println(e);
        }
        System.out.println(arcade);
    }
}
```

exceptions.java

```
/*=====================================================


File                        :  exceptions.java

date                        :  14/4/2025

Author                      :  Benedict Ward

Description                 :  worth 0 marks, just moving all the exceptions here so it
                               doesnt throw errors when uploaded to pass

Possible Exceptions         :  InvalidGameIdException
                               InsufficientBalanceException
                               AgeLimitException
                               InvalidCustomerException

History                     :  14/4/14 - moved all the exceptions here
=====================================================*/



class InvalidGameIdException extends Exception{
    public InvalidGameIdException(String message){
        super(message);
    }
}

class InsufficientBalanceException extends Exception{
    public InsufficientBalanceException(String message){
        super(message);
    }
}

class AgeLimitException extends Exception{
    public AgeLimitException(String message){
        super(message);
    }
}

class InvalidCustomerException extends Exception{
    public InvalidCustomerException(String message){
        super(message);
    }
}
```

Application

Compiler Invocation

```
javac -Xlint:unchecked -Xlint:deprecation -encoding UTF-8 -d
    prepasg1186385049831950855classes Simulation.java ArcadeGame.java CabinetGame.
    java ActiveGame.java VirtualRealityGame.java Customer.java Arcade.java exceptions
    .java
```

Compiler Messages

None.

Application Invocation

```
java Simulation
```

Messages to STDOUT

```
adding arcadegame: VirtualRealityGame{gameId: AVI1USPBNG, pricePerPlay: 0, Name:
 "Virtual UEA Tour", ControlType: HEADSETONLY}
adding arcadegame: VirtualRealityGame{gameId: AVSKVMRB9U, pricePerPlay: 800, Nam
e: "Dance like a Professor", ControlType: null}
adding arcadegame: CabinetGame{gameId: CBGCR27FQM, pricePerPlay: 40, Name: "Reac
tion Test 2000", GiveReward: true}
adding arcadegame: ActiveGame{gameId: AX5YNVUJA9, pricePerPlay: 200, Name: AX5YN
VUJA9, ageRequirement: 16}
adding arcadegame: VirtualRealityGame{gameId: AVLD1ZDNXE, pricePerPlay: 400, Nam
e: "Virtual Petting Zoo", ControlType: HEADSETANDCONTROLLER}
adding arcadegame: ActiveGame{gameId: AL2ETWHG0Q, pricePerPlay: 1000, Name: AL2E
TWHG0Q, ageRequirement: 18}
adding arcadegame: CabinetGame{gameId: CXPVC0DBXU, pricePerPlay: 220, Name: "Clo
ck Crisis", GiveReward: true}
adding arcadegame: CabinetGame{gameId: CNQZPI7G5E, pricePerPlay: 200, Name: "Plu
mber Kart 8", GiveReward: false}
adding arcadegame: VirtualRealityGame{gameId: AV55GWU6PS, pricePerPlay: 350, Nam
e: "Fly Like a Seagull!", ControlType: HEADSETONLY}
adding arcadegame: CabinetGame{gameId: CAPSD7TLC6, pricePerPlay: 200, Name: "Plo
nky Kong", GiveReward: false}
adding arcadegame: ActiveGame{gameId: AHW0HK1F03, pricePerPlay: 80, Name: AHW0HK
1F03, ageRequirement: 3}
adding arcadegame: VirtualRealityGame{gameId: AV9OPS1LRT, pricePerPlay: 1000, Na
me: "VR Paintball", ControlType: FULLBODYTRACKING}
adding arcadegame: CabinetGame{gameId: CPV0HUH0ZH, pricePerPlay: 50, Name: "D'Ar
cy Thompson Pinball", GiveReward: true}
adding arcadegame: ActiveGame{gameId: A4FJTZLIVA, pricePerPlay: 100, Name: A4FJT
ZLIVA, ageRequirement: 12}
adding arcadegame: CabinetGame{gameId: C7S6SYBL8R, pricePerPlay: 120, Name: "CMP
 Chomp Man", GiveReward: false}
adding arcadegame: ActiveGame{gameId: AJFS153KZV, pricePerPlay: 120, Name: AJFS1
53KZV, ageRequirement: 12}
adding arcadegame: VirtualRealityGame{gameId: AVX5KN5T3O, pricePerPlay: 400, Nam
e: "Snowball Fight", ControlType: FULLBODYTRACKING}
adding arcadegame: ActiveGame{gameId: AN234FQD9D, pricePerPlay: 2000, Name: AN23
4FQD9D, ageRequirement: 18}
adding arcadegame: ActiveGame{gameId: AMURG8FXMK, pricePerPlay: 500, Name: AMURG
8FXMK, ageRequirement: 3}
adding arcadegame: CabinetGame{gameId: CQLS3YES0M, pricePerPlay: 140, Name: "Str
eet Wrestler V", GiveReward: false}
adding arcadegame: VirtualRealityGame{gameId: AVPO9GJA1Z, pricePerPlay: 500, Nam
e: "Pickaxe Crafting Simulator", ControlType: HEADSETANDCONTROLLER}
adding arcadegame: ActiveGame{gameId: AD65E3UJQJ, pricePerPlay: 180, Name: AD65E
3UJQJ, ageRequirement: 12}
```

```
[Error]from getCustomer: InvalidCustomerException: No customer found with the ID
 of 900420
could not add £10.0 as we could not find that Id.


==================================================
GamesCo does not take responsibility for any accidents or fits of rage that occu
r on the premises
total number of cabinetgames in this arcade: 8
number of active games in this arcade (not including vr):8
number of virtual reality games in this arcade:7
the richest customer is: Customer{accountID: B473Z4, name: Mantis Toboggan, age:
 72, discounttype: STAFF, balance 20100}
the median price is: £2.00
the total revenue is: £93.16
==================================================
```

Messages to STDERR

None.