**SECTION : 7**

**Practices - Section 7: Date Night at the Arcade**

**Overview**

Tonight is date night at the arcade. After great evening of playing games and winning prizes, you and your date can’t help wondering  
“How are these machines programmed?”. You discuss possible designs on the subway back to campus. You enjoy the rest of the  
night romantically programming your ideas together.  
You’ve made several observations about the arcade. A terminal is used to convert money into game credits. Credits are loaded onto  
plastic game cards. This data is stored in a card’s magnetic strip. Cards may be swiped at any arcade game through the game’s  
magnetic card reader. Games subtract credits from a card, but awards tickets. Tickets are also stored on a card’s magnetic strip.  
Tickets may be exchanged for prizes at the terminal. The terminal is also used to check a card’s credit balance and ticket count, and to  
transfer credits or tickets between cards.

**Tasks**

Write a Java program that models the properties, behaviors, and interactions of objects at the arcade. You’ll also need a test class that  
contains a main method. Use the main method to model actions that would drive the program such as object instantiations and card  
swipes. All fields must be private. Provide getter and any necessary setter methods.

**Cards**

The magnetic strip on game cards offers limited storage space and zero computing power. Cards store information about their current  
credit balance, ticket balance, and card number. Neither balance should ever be negative. Individual cards are incapable of performing  
calculations, including simple addition, or realizing that their balances could go negative.  
Every card is created with a unique integer identification number. Although each individual card is incapable of simple addition, it’s still  
possible to perform calculations with properties that belong to all cards.

Copyright © 2022, Oracle and/or its affiliates. Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

**PROGRAM :**

public class Card

{

private int cardNumber;

private int creditBalance;

private int ticketBalance;

public Card(int cardNumber)

{

this.cardNumber = cardNumber;

this.creditBalance = 0;

this.ticketBalance = 0;

}

public int getCardNumber()

{

return cardNumber;

}

public int getCreditBalance()

{

return creditBalance;

}

public int getTicketBalance()

{

return ticketBalance;

}

public void addCredits(int credits)

{

this.creditBalance += credits;

}

public void addTickets(int tickets)

{

this.ticketBalance += tickets;

}

public boolean deductCredits(int credits)

{

if (this.creditBalance >= credits)

{

this.creditBalance -= credits;

return true;

}

else

{

return false;

}

}

public boolean deductTickets(int tickets)

{

if (this.ticketBalance >= tickets)

{

this.ticketBalance -= tickets;

return true;

}

else

{

return false;

}

}

public void transferCredits(Card otherCard, int credits)

{

if (deductCredits(credits))

{

otherCard.addCredits(credits);

}

else

{

System.out.println("Insufficient credits to transfer.");

}

}

public void transferTickets(Card otherCard, int tickets)

{

if (deductTickets(tickets))

{

otherCard.addTickets(tickets);

}

else

{

System.out.println("Insufficient tickets to transfer.");

}

}

}

**Games**

Games require a certain number of credits to be played. Each game is equipped with a magnetic card reader and LCD display.  
Swiping a card reduces its credit balance, but awards a random, non-negative number of tickets. Print the card number, number of  
tickets won, along with the new total. Print a message if a card has insufficient credits to play a game.  
The “Win Random Tickets Game!” is actually a terrible game. You’re welcome to create something more complex, but it’s not  
necessary for this assignment.

**PROGRAM :**

import java.util.Random;

public class Game

{

private String name;

private int requiredCredits;

public Game(String name, int requiredCredits)

{

this.name = name;

this.requiredCredits = requiredCredits;

}

public void play(Card card)

{

if (card.deductCredits(requiredCredits))

{

int ticketsWon = new Random().nextInt(10);

card.addTickets(ticketsWon);

System.out.println("Card #" + card.getCardNumber() + " won " + ticketsWon + " tickets. New ticket balance: " + card.getTicketBalance());

}

else

{

System.out.println("Card #" + card.getCardNumber() + " has insufficient credits to play " + name + ".");

}

}

}

**Prize Categories**

Each prize category has a name, number of tickets required to earn that prize, and a count of how many items of this category remain  
in a terminal. Prizes know nothing about the terminal they belong to.

**PROGRAM :**

public class PrizeCategory

{

private String name;

private int requiredTickets;

private int count;

public PrizeCategory(String name, int requiredTickets, int count)

{

this.name = name;

this.requiredTickets = requiredTickets;

this.count = count;

}

public String getName()

{

return name;

}

public int getRequiredTickets()

{

return requiredTickets;

}

public int getCount()

{

return count;

}

public boolean awardPrize()

{

if (count > 0)

{

count--;

return true;

}

else

{

return false;

}

}

}

**Terminals**

Each terminal contains a magnetic card reader. A terminal accepts money which is converted to credits on a card. Money is accepted  
as whole numbers. Credits are awarded at a rate of 2 credits for every $1. Players may use a Terminal to check their card’s balances.  
Include the card’s number in this printout. All or just a portion of credits or tickets may be transferred between cards. Always print a  
card’s balances when either credits or tickets are accessed through a terminal. Finally, tickets may be exchanged at terminals for  
prizes. Print an error message if a card has insufficient tickets or if the terminal is out of a particular prize type. Print when a prize is  
awarded and the remaining number of that prize type in the terminal. A terminal offers 3 categories of prizes.

**PROGRAM :**

public class Terminal

{

private PrizeCategory[] prizes;

public Terminal(PrizeCategory[] prizes)

{

this.prizes = prizes;

}

public void loadCredits(Card card, int dollars)

{

int credits = dollars \* 2;

card.addCredits(credits);

System.out.println("Loaded " + credits + " credits to Card #" + card.getCardNumber() + ". New credit balance: " + card.getCreditBalance());

}

public void checkBalances(Card card)

{

System.out.println("Card #" + card.getCardNumber() + " - Credits: " + card.getCreditBalance() + ", Tickets: " + card.getTicketBalance());

}

public void transferCredits(Card fromCard, Card toCard, int credits)

{

fromCard.transferCredits(toCard, credits);

checkBalances(fromCard);

checkBalances(toCard);

}

public void transferTickets(Card fromCard, Card toCard, int tickets)

{

fromCard.transferTickets(toCard, tickets);

checkBalances(fromCard);

checkBalances(toCard);

}

public void redeemPrize(Card card, String prizeName)

{

for (PrizeCategory prize : prizes)

{

if (prize.getName().equals(prizeName))

{

if (card.deductTickets(prize.getRequiredTickets()))

{

if (prize.awardPrize())

{

System.out.println("Awarded " + prizeName + ". Remaining: " + prize.getCount());

}

else

{

System.out.println("Out of " + prizeName + ".");

}

}

else

{

System.out.println("Insufficient tickets for " + prizeName + ".");

}

return;

}

}

System.out.println("Prize not found.");

}

}

**Main Method**

Instantiate 2 cards and whatever other objects might be necessary to test your program.  
• Load credits onto each card.  
• Play a bunch of game using both cards.  
• Transfer the balance of credits and tickets from Card 1 to Card 2.  
• Request prizes using Card 2.  
• Try to play a game and request a prize using Card 1.  
• Perform whatever other actions might be necessary to test your program

**PROGRAM :**

public class ArcadeTest

{

public static void main(String[] args)

{

PrizeCategory smallPrize = new PrizeCategory("Small Prize", 10, 5);

PrizeCategory mediumPrize = new PrizeCategory("Medium Prize", 20, 3);

PrizeCategory largePrize = new PrizeCategory("Large Prize", 30, 2);

Terminal terminal = new Terminal(new PrizeCategory[]{smallPrize, mediumPrize, largePrize});

Card card1 = new Card(1001);

Card card2 = new Card(1002);

terminal.loadCredits(card1, 5); // $5 = 10 credits

terminal.loadCredits(card2, 10); // $10 = 20 credits

Game game = new Game("Win Random Tickets Game!", 3);

game.play(card1);

game.play(card2);

game.play(card1);

terminal.transferCredits(card1, card2, 5);

terminal.transferTickets(card1, card2, 2);

terminal.redeemPrize(card2, "Small Prize");

game.play(card1);

terminal.redeemPrize(card1, "Large Prize");

terminal.checkBalances(card1);

terminal.checkBalances(card2);

}

}