**JAVA PRACTICE**

**SECTION 7**

import java.util.Random;

class ArcadeCard {

private int cardNumber;

private int creditBalance;

private int ticketBalance;

public ArcadeCard(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) {

creditBalance += credits;

}

public void subtractCredits(int credits) {

if (creditBalance >= credits) {

creditBalance -= credits;

} else {

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

}

}

public void addTickets(int tickets) {

ticketBalance += tickets;

}

public void subtractTickets(int tickets) {

if (ticketBalance >= tickets) {

ticketBalance -= tickets;

} else {

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

}

}

}

// Game class

class Game {

private String name;

private int creditsRequired;

private int ticketBalance;

public Game(String name, int creditsRequired) {

this.name = name;

this.creditsRequired = creditsRequired;

this.ticketBalance = 0;

}

public String getName() {

return name;

}

public int getCreditsRequired() {

return creditsRequired;

}

public int getTicketBalance() {

return ticketBalance;

}

public void play(ArcadeCard card) {

if (card.getCreditBalance() >= creditsRequired) {

card.subtractCredits(creditsRequired);

Random random = new Random();

int ticketsWon = random.nextInt(10);

card.addTickets(ticketsWon);

ticketBalance += ticketsWon;

System.out.println("Card " + card.getCardNumber() + " played " + name + " and won " + ticketsWon + " tickets.");

} else {

System.out.println("Card " + card.getCardNumber() + " does not have enough credits to play " + name + ".");

}

}

}

// PrizeCategory class

class PrizeCategory {

private String name;

private int ticketsRequired;

private int itemCount;

public PrizeCategory(String name, int ticketsRequired, int itemCount) {

this.name = name;

this.ticketsRequired = ticketsRequired;

this.itemCount = itemCount;

}

public String getName() {

return name;

}

public int getTicketsRequired() {

return ticketsRequired;

}

public int getItemCount() {

return itemCount;

}

public void decreaseItemCount() {

if (itemCount > 0) {

itemCount--;

} else {

System.out.println("No more items left in category " + name);

}

}

}

// Terminal class

class Terminal {

private int creditRate;

private PrizeCategory[] prizeCategories;

public Terminal(int creditRate, PrizeCategory[] prizeCategories) {

this.creditRate = creditRate;

this.prizeCategories = prizeCategories;

}

public void insertMoney(int money, ArcadeCard card) {

int credits = money \* creditRate;

card.addCredits(credits);

System.out.println("Inserted $" + money + " into Card " + card.getCardNumber() + ". Added " + credits + " credits.");

}

public void checkCardBalance(ArcadeCard card) {

System.out.println("Card " + card.getCardNumber() + " has " + card.getCreditBalance() + " credits and " + card.getTicketBalance() + " tickets.");

}

public void transferCredits(ArcadeCard fromCard, ArcadeCard toCard, int credits) {

if (fromCard.getCreditBalance() >= credits) {

fromCard.subtractCredits(credits);

toCard.addCredits(credits);

System.out.println("Transferred " + credits + " credits from Card " + fromCard.getCardNumber() + " to Card " + toCard.getCardNumber() + ".");

} else {

System.out.println("Card " + fromCard.getCardNumber() + " does not have enough credits to transfer.");

}

}

public void requestPrize(ArcadeCard card, int categoryIndex) {

if (categoryIndex >= 0 && categoryIndex < prizeCategories.length) {

PrizeCategory category = prizeCategories[categoryIndex];

if (card.getTicketBalance() >= category.getTicketsRequired()) {

if (category.getItemCount() > 0) {

card.subtractTickets(category.getTicketsRequired());

category.decreaseItemCount();

System.out.println("Card " + card.getCardNumber() + " redeemed a prize from category " + category.getName() + ".");

System.out.println("Remaining " + category.getName() + " prizes: " + category.getItemCount());

} else {

System.out.println("No more prizes left in category " + category.getName() + ".");

}

} else {

System.out.println("Card " + card.getCardNumber() + " does not have enough tickets to redeem a prize from category " + category.getName() + ".");

}

} else {

System.out.println("Invalid prize category index.");

}

}

}

// Main class

public class ArcadeSimulation {

public static void main(String[] args) {

// Initialize cards

ArcadeCard card1 = new ArcadeCard(1);

ArcadeCard card2 = new ArcadeCard(2);

// Add initial credits

card1.addCredits(10);

card2.addCredits(20);

// Initialize games

Game game1 = new Game("Game 1", 5);

Game game2 = new Game("Game 2", 8);

// Play games

game1.play(card1);

game2.play(card2);

// Initialize prize categories

PrizeCategory[] prizeCategories = {

new PrizeCategory("Stuffed Animal", 50, 10),

new PrizeCategory("Action Figure", 100, 5),

new PrizeCategory("Puzzle", 150, 2)

};

// Initialize terminal

Terminal terminal = new Terminal(2, prizeCategories);

// Transfer credits

terminal.transferCredits(card1, card2, 5);

// Request prizes

terminal.requestPrize(card2, 0);

game1.play(card1);

terminal.requestPrize(card1, 1);

// Check balances

terminal.checkCardBalance(card1);

terminal.checkCardBalance(card2);

}

}