Java Foundations

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.

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;
```

```java
    }

    public void addCredits(int credits) {
        if (credits > 0) {
            this.creditBalance += credits;
        }
    }

    public void subtractCredits(int credits) {
        if (credits > 0 && this.creditBalance >= credits) {
            this.creditBalance -= credits;
        }
    }

    public void addTickets(int tickets) {
        if (tickets > 0) {
            this.ticketBalance += tickets;
        }
    }

    public void subtractTickets(int tickets) {
        if (tickets > 0 && this.ticketBalance >= tickets) {
            this.ticketBalance -= tickets;
        }
    }
}
```

GAME CLASS:

Program:

```java
import java.util.Random;

public class Game {
    private String name;
    private int creditsRequired;

    public Game(String name, int creditsRequired) {
        this.name = name;
        this.creditsRequired = creditsRequired;
    }

    public String getName() {
        return name;
    }

    public int getCreditsRequired() {
        return creditsRequired;
    }

    public void play(Card card) {
        if (card.getCreditBalance() >= creditsRequired) {
            card.subtractCredits(creditsRequired);
            Random rand = new Random();
            int ticketsWon = rand.nextInt(11); // Random tickets between 0 and 10
            card.addTickets(ticketsWon);
            System.out.println("Card Number: " + card.getCardNumber() + ", Tickets Won: " + ticketsWon +
", New Ticket Total: " + card.getTicketBalance());
        } else {
            System.out.println("Card Number: " + card.getCardNumber() + " has insufficient credits to play
" + name);
        }
    }
}
```

Prize class:

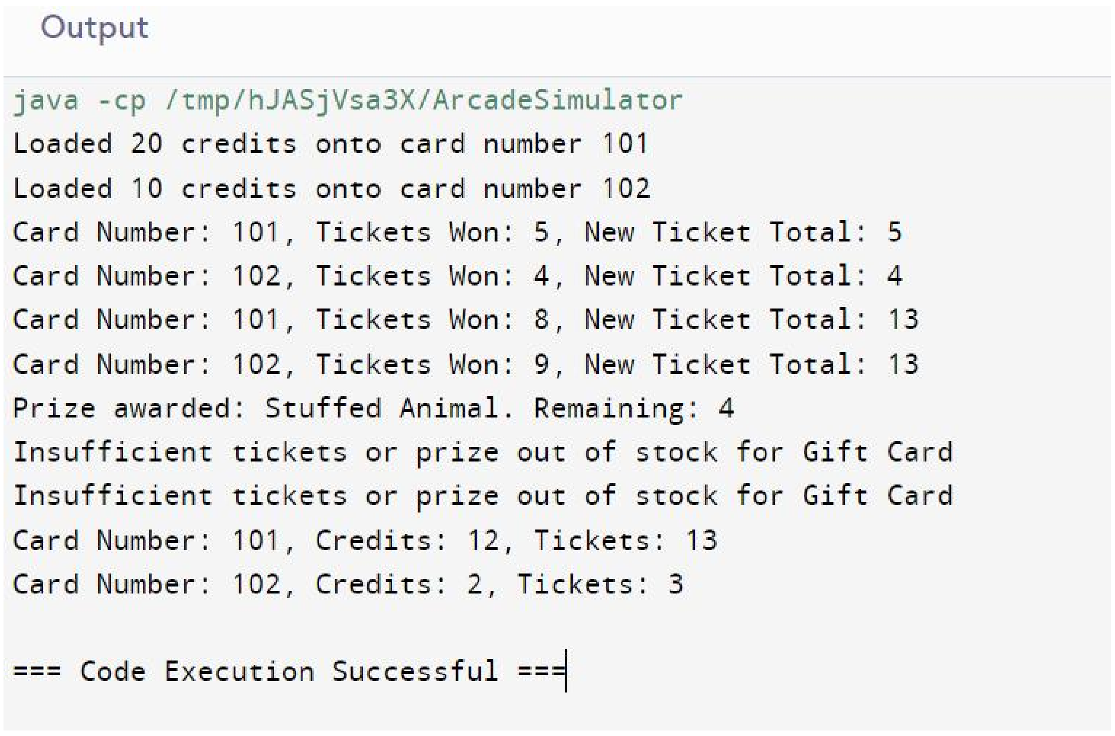
```
public class Prize {
    private String name;
    private int ticketsRequired;
    private int quantity;

    public Prize(String name, int ticketsRequired, int quantity) {
        this.name = name;
        this.ticketsRequired = ticketsRequired;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public int getTicketsRequired() {
        return ticketsRequired;
    }

    public int getQuantity() {
        return quantity;
    }

    public boolean redeem(Card card) {
        if (card.getTicketBalance() >= ticketsRequired && quantity > 0) {
            card.subtractTickets(ticketsRequired);
            quantity--;
            System.out.println("Prize awarded: " + name + ". Remaining: " + quantity);
            return true;
        } else {
            System.out.println("Insufficient tickets or prize out of stock for " + name);
            return false;
        }
    }
}
```

Terminal class:

```
public class Terminal {
    public void loadCredits(Card card, int money) {
        int credits = money * 2; // 2 credits for every $1
        card.addCredits(credits);
        System.out.println("Loaded " + credits + " credits onto card number " + card.getCardNumber());
    }

    public void checkBalances(Card card) {
        System.out.println("Card Number: " + card.getCardNumber() + ", Credits: " +
card.getCreditBalance() + ", Tickets: " + card.getTicketBalance());
    }
}
```

ArcadeSimulator Class (Main Class):

```
public class ArcadeSimulator {
    public static void main(String[] args) {
        // Instantiate cards
        Card card1 = new Card(101);
        Card card2 = new Card(102);

        // Instantiate terminal and load credits
        Terminal terminal = new Terminal();
        terminal.loadCredits(card1, 10); // $10 -> 20 credits
        terminal.loadCredits(card2, 5);   // $5 -> 10 credits

        // Instantiate games
        Game game1 = new Game("Win Random Tickets Game", 5);
        Game game2 = new Game("Another Game", 3);

        // Play games
        game1.play(card1);
        game1.play(card2);
        game2.play(card1);
        game2.play(card2);

        // Instantiate prizes
        Prize prize1 = new Prize("Stuffed Animal", 10, 5);
        Prize prize2 = new Prize("Gift Card", 15, 2);

        // Redeem prizes
        prize1.redeem(card2);
        prize2.redeem(card1);
        prize2.redeem(card2);

        // Check balances
        terminal.checkBalances(card1);
        terminal.checkBalances(card2);
    }
}
```


Output

```
java -cp /tmp/hJASjVsa3X/ArcadeSimulator
Loaded 20 credits onto card number 101
Loaded 10 credits onto card number 102
Card Number: 101, Tickets Won: 5, New Ticket Total: 5
Card Number: 102, Tickets Won: 4, New Ticket Total: 4
Card Number: 101, Tickets Won: 8, New Ticket Total: 13
Card Number: 102, Tickets Won: 9, New Ticket Total: 13
Prize awarded: Stuffed Animal. Remaining: 4
Insufficient tickets or prize out of stock for Gift Card
Insufficient tickets or prize out of stock for Gift Card
Card Number: 101, Credits: 12, Tickets: 13
Card Number: 102, Credits: 2, Tickets: 3

=== Code Execution Successful ===
```