PROJECT-7

- ❖ 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.

CODE:

```
package arcade;
public class card {
	private int id;
	private double credits;
	private int tickets;
	public card(int id) {
	  this.id = id;
	  this.credits = 0;
	  this.tickets = 0;
	}
	public void loadCredits(double amount) {
	  credits += amount;
	}
	public boolean playGame(game game) {
	  if (credits >= game.getCost()) {
	    credits -= game.getCost();
	    tickets += game.getTickets();
	    return true;
	  }
	  return false;
	}
	public void transferTo(card other) {
	  other.credits += this.credits;
	  other.tickets += this.tickets;
	  this.credits = 0;
	  this.tickets = 0;
	}

	public boolean requestPrize(prize prize) {
	  if (tickets >= prize.getTicketCost()) {
```

```
            tickets -= prize.getTicketCost();
            return true;
        }
        return false;
    }

    public double getCredits() {
        return credits;
    }

    public int getTickets() {
        return tickets;
    }
}
```

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.

CODE:

```
package arcade;
public class game {
    private String name;
    private double cost;
    private int tickets;
    public game(String name, double cost, int tickets) {
        this.name = name;
        this.cost = cost;
        this.tickets = tickets;
    }
    public double getCost() {
        return cost;
    }
    public int getTickets() {
        return tickets;
    }
}
```

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.

```
package arcade;

public class Prize {
   private String name;
   private int ticketCost;

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

   public int getTicketCost() {
      return ticketCost;
   }
}
```

ARCADE:

```
package arcade;

public class ArcadeTest {
   public static void main(String[] args) {
      card card1 = new card(1);
      card card2 = new card(2);

      // Load credits onto each card
      card1.loadCredits(50.0);
      card2.loadCredits(30.0);

      // Instantiate games
      game game1 = new game("Space Invaders", 10.0, 5);
      game game2 = new game("Pac-Man", 5.0, 3);

      // Play games using both cards
      card1.playGame(game1);
      card1.playGame(game2);
      card2.playGame(game1);

      // Transfer balance from card1 to card2
      card1.transferTo(card2);

      // Instantiate prizes
      Prize prize1 = new Prize("Stuffed Animal", 10);
      Prize prize2 = new Prize("Toy Car", 15);

      // Request prizes using card2
      card2.requestPrize(prize1);
      card2.requestPrize(prize2);
```

```
        // Try to play a game and request a prize using card1
        boolean card1Play = card1.playGame(game2);
        boolean card1Prize = card1.requestPrize(prize1);

        // Display results
        System.out.println("Card 1: " + card1.getCredits() + " credits, " + card1.getTickets() + " tickets");
        System.out.println("Card 2: " + card2.getCredits() + " credits, " + card2.getTickets() + " tickets");
        System.out.println("Card 1 tried to play a game: " + (card1Play ? "Success" : "Failed"));
        System.out.println("Card 1 tried to request a prize: " + (card1Prize ? "Success" : "Failed"));
    }
}
```