

Sample Solution

```
public class ArcadeCard
{
    private static int cardsIssued = 0;
    private static int totalSpent = 0;

    private String owner;
    private int tokens;
    private int cardNumber;

    public ArcadeCard(String owner, int tokens)
    {
        this.owner = owner;
        this.tokens = tokens;
        cardsIssued++;
        cardNumber = cardsIssued;
    }

    public ArcadeCard(String owner)
    {
        this.owner = owner;
        tokens = 5;
        cardsIssued++;
        cardNumber = cardsIssued;
    }

    public boolean play(int cost)
    {
        if (tokens >= cost)
        {
            tokens -= cost;
            totalSpent += cost;
            return true;
        }
        return false;
    }

    public boolean play()
    {
        return play(2);
    }

    public void addTokens(int amount)
    {
        tokens += amount;
    }

    public int getTokens()
    {
        return tokens;
    }

    public int getCardNumber()
    {
        return cardNumber;
    }

    public static int getTotalSpent()
    {
        return totalSpent;
    }
}
```

Scoring Guidelines (9 points)

1. Declares the class header and private instance variables for the token count and the card number. The owner is optional, since nothing reads it.
2. Declares private static variables for the card count and the total spent.
3. Two-argument constructor: initializes the tokens from the parameter.
4. One-argument constructor: initializes the tokens to 5.
5. Both constructors increment the card count and assign the card number correctly.
6. `play(int cost)`: checks for enough tokens and returns the correct boolean.
7. `play(int cost)`: on success, updates both the card's tokens and the static total. On failure, changes neither.
8. `play()`: calls `play(2)` or duplicates that logic correctly, and returns the result.
9. `addTokens`, `getTokens`, and `getCardNumber` are correct, and `getTotalSpent` is declared static.

Common mistakes to watch for

- Assigning `cardNumber` before incrementing, which numbers the cards starting at 0.
- Forgetting static on `getTotalSpent`. The table's `ArcadeCard.getTotalSpent()` call is the clue.
- Adding the cost to the total even when the play fails.
- Making `totalSpent` an instance variable, so each card tracks only its own spending.