# Exercises: Enumerations and Attributes

This document defines the exercises for ["C# OOP Basics" course @ Software University](https://softuni.bg/trainings/1375/java-basics-oop-june-2016). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/).

## Card Suit

Create an enumeration type that has as its constants the four suits of a deck of playing cards (Clubs, Hearts, Diamonds, Spades) Iterate over the values of the enumeration type and print all ordinal values and names.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Card Suits | Card Suits:  Ordinal value: 0; Name value: Clubs  Ordinal value: 1; Name value: Diamonds  Ordinal value: 2; Name value: Hearts  Ordinal value: 3; Name value: Spades |

## Card Rank

Create an enumeration type that has as its constants the fourteen ranks of a deck of playing cards (Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King) Iterate over the values of the enumeration type and print all ordinal values and names.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Card Ranks | Card Ranks:  Ordinal value: 0; Name value: Ace  Ordinal value: 1; Name value: Two  Ordinal value: 2; Name value: Three  Ordinal value: 3; Name value: Four  Ordinal value: 4; Name value: Five  Ordinal value: 5; Name value: Six  Ordinal value: 6; Name value: Seven  Ordinal value: 7; Name value: Eight  Ordinal value: 8; Name value: Nine  Ordinal value: 9; Name value: Ten  Ordinal value: 10; Name value: Jack  Ordinal value: 11; Name value: Queen  Ordinal value: 12; Name value: King |

## Card Power

Create **class** **Card** that holds **Rank** and **Suit**. Create a program that generates a deck of cards which have a power. The power of a card is calculated by adding the power of its rank plus the power of its suit.

Rank powers are as follows: (Ace - 14, Two - 2, Three - 3, Four - 4, Five - 5, Six - 6, Seven - 7, Eight - 8, Nine - 9, Ten - 10, Jack - 11, Queen - 12, King - 13).

Suit powers are as follows: (Clubs - 0, Diamonds - 13, Hearts - 26, Spades - 39).

You will get a command consisting of two lines. On the first line you will receive the Rank of the card and on the second line you will get the suit of the card.

Print the output in the format "Card name: ACE of SPADES; Card power: 53".

### Note

Try using the enumeration types you have created in the previous problems but extending them with constructors and methods. To get the card power cast to integer Rank and Suit and add them together.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Two  Clubs | Card name: Two of Clubs; Card power: 2 |
| Ace  Spades | Card name: Ace of Spades; Card power: 53 |

## Card ToString()

If you haven’t done it already override the **ToString()** of your **Card** class you've created earlier. Make it so it returns the same information as before e.g. in format:

"Card name: {Rank} of {Suit}; Card power: {Card power}"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Two  Clubs | Card name: Two of Clubs; Card power: 2 |
| Ace  Spades | Card name: Ace of Spades; Card power: 53 |

## Card CompareTo()

As your cards have power you can safely add a functionality for comparing them. Try using the already available interface **IComparable<T>** and override the **CompareTo()** method.

Read two cards from the console and print the greater of the two. In the given format:

"Card name: {Rank} of {Suit}; Card power: {Card power}"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Two  Clubs  Ace  Spades | Card name: Ace of Spades; Card power: 53 |

## Custom Enum Attribute

Create a custom attribute **TypeAttribute** that can be applied to classes and can be accessed at runtime. The **TypeAttribute** elements should contain **type,** **category** and **description** as parameters. Apply the attribute to both enumeration types you have created for the previous problems (Rank and Suit). Provide them these exact values:

Rank:

* type = "Enumeration"
* category = "Rank"
* description = "Provides rank constants for a Card class."

Suit:

* type = "Enumeration"
* category = "Suit"
* description = "Provides suit constants for a Card class."

Create a program which gets the description of an enumeration type by a given rank.

### Note

Try using the [**typeof(TypeAttribute).GetCustomAttributes()**](https://msdn.microsoft.com/en-us/library/system.type.getcustomattributes(v=vs.110).aspx) method.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Rank | Type = Enumeration, Description = Provides rank constants for a Card class. |

## Deck of Cards

Create a program that generates all cards of a card playing deck. First print the clubs, starting from the Ace, ending with a King. Continue with the same cards from Hearts, Diamonds and Spades. Print them in the format given below.

### Note

Try using the enumeration types you have created in the previous problems.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Card Deck | Ace of Clubs  Two of Clubs  Three of Clubs  Four of Clubs  Five of Clubs  ...  ...  King of Spades |

## Card Game

Simulate a card game in which you have two players. Each player has a hand of five cards. The winning player is the player which holds the highest powered card in his hand.

Rank powers are as follows: (Ace - 14, Two - 2, Three - 3, Four - 4, Five - 5, Six - 6, Seven - 7, Eight - 8, Nine - 9, Ten - 10, Jack - 11, Queen - 12, King - 13).

Suit powers are as follows: (Clubs - 0, Diamonds - 13, Hearts - 26, Spades - 39).

### Input

On the first two lines you will get the names of the players.

On the next lines, you should **read cards** from the console in the format **{Ace of Clubs}** for a certain player until he has **exactly 5 cards in his hand**. If he receives a card that is not in the deck, you should print "Card is not in the deck." If he receives an invalid card name, for example "spades of Ace", print "No such card exists."

### Output

Print the name of the winner and his winning card in the format "{Player name} wins with {Card name}."

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Ivo  Gosho  Queen of Diamonds  King of Diamonds  **Ace of Hearts**  Ace of Hearts  spades of Ace  Two of Hearts  Three of Hearts  Four of Hearts  Five of Hearts  Six of Hearts  Seven of Hearts  Eight of Hearts | Card is not in the deck.  No such card exists.  Ivo wins with Ace of Hearts. | Player Ivo receives cards (in orange) from the deck, until he has exactly five of them.  When he is given ACE of HEARTS for a second time, error message is printed and his hand stays the same size.  When a card with invalid name is given, error message is printed and his hand stays the same size.  When Ivo's hand has 5 cards, Gosho starts receiving cards from the deck.  When Gosho has 5 cards, the hands are evaluated and one of the players wins. |