CS 343 – Structure of Programming Languages
Winter 2011, March 17, 2011

Programming Assignment in Ruby
Due Date: Monday, March 28, 2011

Goals

- Implement classes in Ruby
- Add methods to a Ruby supplied class (the Array class)
- Exception handling

Description

You are provided with a Ruby source file called "`Project4.rb`" that is incomplete. You can download this file from Blackboard. This source file contains three unfinished classes (`Card`, `Deck`, and the `Array` class from Ruby API) and a global function called `main`. Complete the `Card` and `Deck` classes. You will also need to add two new methods to the `Array` class. The `main` function contains code to test the functionality you are asked to implement in classes `Card`, `Deck`, and `Array`.

Card Class

Method	Description
`initialize(rank,suit)`	• Initializes the instance variables `@rank` and `@suit` using the values of parameters `rank` and `suit`. • DO NOT set the `@rank` and `@suit` directly in this method. Use the setter methods `rank=` and `suit=` to initialize the instance variables.
`rank=(rank)`	• Setter/writer method that sets the value of instance variable `@rank` with the value of parameter `rank`. • Raises the `ArgumentError` exception if the parameter is not valid.
`suit=(suit)`	• Setter/writer method that sets the value of instance variable `@suit` with the value of parameter `suit`. • Raises the `ArgumentError` exception if the parameter is not valid.
`to_s`	• Returns a string representation of a `Card` object in this format: rank followed by suit. Examples: "2d", "Ah", "Ts" • Rank information must be in uppercase if it is a letter. • Suit information must be in lowercase.

Deck Class

Method	Description
`initialize()`	• Creates an instance variable called `@pack` – an array that can store 52 `Card` objects. • Instantiates 52 `Card` objects and stores them in `@pack` variable.
`shuffle`	• Shuffles (randomly arranges) the cards in the array `@pack`. See the `Array` class for a method you can use to do this.
`empty?`	• Return `true` if the number of cards in the deck is zero, `false` otherwise.
`deal(n=1)`	• If `n` is 1 (default value), returns a card from the top (end of the array `@pack`) of the deck. If `n` > = 2, it return an `Array` of cards of that size. If there are not enough cards, it should return as many (the remaining cards) cards as possible. If the deck is empty, it should return `nil`. If the value of `n` is not an `Integer` type or < 1, it should return `nil`.
`size`	• Returns the number of cards in the deck.
`to_s`	• Returns the string representation of the deck (`@pack.to_s`)

Array Class

Method	Description
`^(other)`	• Returns an `Array` that represents the *symmetric difference* of self (this) array and other array. • The symmetric difference of two sets is the set of elements which are in either of the sets, but not in their intersection. It can be expressed as the union of the two sets, minus their intersection. • See the `Array` class for methods you can use to do this.
`**(other)`	• Returns an `Array` that represents the Cartesian product of self (this) array and other array. • The Cartesian product of two sets is defined as the set of all pairs such that the first element is in the first set and the second element is in the second set.

Deliverables

1. Upload the finished **Project4.rb** on Blackboard.
    - I will use the submission date/time on Blackboard as your official submission date/time.
    - It is your responsibility to make sure the submission on Blackboard went through successfully.
2. Access to the assignment will be turned off end of day on Thursday, March 31st (three days after the due date with a late penalty of 10 points day).