CS112 - Spring 2024
Lab07
Instructor: Paul Haskell

INTRODUCTION

In this week-long lab, you will work with several of the new Java capabilities we discussed this week: UTF-16, keyboard input, class Random, etc. There are four parts to the lab, and the fourth one is big, so please manage your time well.

UTF-16 and UTF-8

Let's start with a really short program, **GetUtf.java** . Please cut and paste this character ࢢ into your Java program. Print out its UTF-16 coded value as an integer. That's it, as long as your editor lets you paste this character in nicely. (If not, try writing and running your program in Eclipse.) Does the character look the same in your editor as in this PDF file?

For fun, if you want, you can try pasting in other characters to see their UTF-16 coded values. But don't turn that in!

Objects and References

For this project, you will look at and run code that illustrates the workings of references vs objects. There is no coding to code. However, it is tricky—but important—to understand what is going on. You probably want to draw on a piece of paper all of the objects and the references to them.

Copy the program **ObjsAndRefs.java** from the **CourseInfo/Lab07** directory into a **Lab07** directory in your personal repository. Build and run it.

After you run the program and see its output, your mission is to explain the output. In particular, in a file called **explanation.txt** , write a few sentences explaining why you get the results you do. You should explain why the results with `class Integer` and `class MyInt` are different.

Keyboard Input

Create a program called **EightLines.java**. This program should read in exactly eight lines of text (as returned by `Scanner.nextLine()`) from the keyboard input. Each line may contain 0 or more words. (A "word" is any set of non-space characters, separated from other words on the same line by one or more spaces.) Each input word should be printed alone on a line.

What kinds of test cases can you think of?

- Lines with 0, 1, or more than 1 words

- Fewer than 8, exactly 8, or more than 8 input lines. How do you detect if the input contains fewer than 8 lines? What method should you use?

If an input line contains no words, what should the program do? The specification says only to print words at the input, so such lines should be "counted as lines" but should not yield any output.

How do you want to get each word from each line? You could use `String.split()`, or you could make a new `Scanner` for the line, reading from the `String` that contains the line.

Deck of Cards

This is a bigger programming exercise than we have had so far.

First, please create a `class Card` that represents a single card from a standard 52-card deck of playing cards.

Each card has a value (2,3,4,5,6,7,8,9,10,11,12,13,14) and a suit (Spades, Hearts, Clubs, Diamonds).

Your class should have two constructors:

- One that takes an `int` and a `String`. The integer (2 through 14) is used to set the card value. (2 through 10 are obvious. Jack has value 11, Queen has value 12, King has 13, and Ace has 14). The `String` sets the suit. To set the suit, look at the first letter of the argument, ignoring upper/lowercase, and if it is {s,h,c,d} then set the suit accordingly (Spades, Hearts, Clubs, Diamonds).
- Another constructor takes a `String` and parses it to set the card value and suit. If the first character in the string is 2 through 9 or J, Q, K, or A (ignore upper/lowercase), then set the card value in the obvious way. If the first character is '1', then the second character must be '0' to indicate a card value of 10. The following character should be {s,h,c,d} to specify the suit. Accept either uppercase or lowercase letters to set the suit. Any following characters in the input parameter can be ignored.

- If there is any error in the input to the constructor, the card value should be set to 0 and the suit should be set to "ERROR". What are errors? Values not in the range from 2 through 14 (or not 'j', 'q', 'k', or 'a' for the second constructor), or suit letters other than s, h, c, d.

In addition to the constructors, please add the following methods:

- `Value()`: returns the card's value as an integer in the range from 2 through 14.
- `Suit()`: returns the card's suit as a `String`. The suit should be represented as a single capital letter: "S", "H", "C", or "D"
- A method called `toString()` that returns a `String` showing the value and then the suit, with no space between them. However, instead of printing 11 for Jack, 12 for Queen, etc instead print "J" for Jack, "Q" for Queen, "K" for King, and "A" for Ace. For example, this method should print "10D", "QC", etc.

Class Deck

Your class Deck will store 52 cards: 13 cards for each suit, with values from 2 through 14 (or 2 through Ace, if you prefer). To store these, you will use an array of `Card` objects

Here, you will want code that looks something like:

```
Card allMyCards[] = new Card[52];
```

This creates an array called `allMyCards` that has storage for 52 Cards. The storage slots in the array are not initialized yet, but you can access them by index, e.g.

```
allMyCards[0], allMyCards[1], … allMyCards[51]
```

To initialize each array entry, in the `Deck` constructor, you use the new operator, for example:

```
allMyCards[0] = new Card(2, "spades");
```

Ok, now you have instructions on how to create an empty array and on how to initialize each entry in the array. But you probably don't want to copy and paste 52 different lines of code to initialize each entry. (For an array with 52 entries, the indices are 0 through 51.) Instead, use a loop, or maybe a few of them: one to set the suit and one to set the card value. Or you could have separate loops for each suit.

Other methods in class Deck

- Add a `toString()` method. This method returns a `String` that contains each card value in the deck, with a space (" ") between the individual card values.
- `void shuffle()` – this method shuffles all the cards in the deck randomly. Be sure you "shuffle" rather than just choosing card values randomly. After shuffling, the cards in the deck should have a random order, but there should still be exactly one of every card value and suit.

Details of this Assignment

Please put your `class Card` and `class Deck` into a file called **MyCardDeck.java**. Of course, you also need a `class MyCardDeck` that will hold the `main()` function. In `main()` do the following:

- If there is one or more command-line argument to the program, initialize a `Card` with the first argument, passed to the one-String constructor. Then print the `Card` value
  ```
  System.out.println(yourCard);
  ```
  and exit.
- If there are zero arguments to the program, create an object of type `Deck`, shuffle it, print out the entire deck, then exit.

Reminder

Print exactly what was requested. Put useful comments into your code. Design and organize your code so that it is easy for others to understand.

Put your files into the **Lab07** directory your **MyWork** directory, and remember to push your **Lab07** to GitHub before the deadline. This assignment must be turned in before Sunday Feb 18 at 11:59pm.

Conclusion

This is a bigger lab, with a lot to code and understand. Congratulations for getting through it!

Grading Rubric

GetUtf.java is worth 5 points if it prints the correct encoded value.

The **explanation.txt** document is worth 10 points:

- One point if it has the correct name, is located in the correct directory, and is an actual text file rather than a Word file or PDF, etc.
- Nine points for the clarity and accuracy of your explanation.

EightLines.java is worth 15 points, 3 points for each of 5 test cases.

MyCardDeck.java is worth 40 points:

- Two points if it compiles
- 14 points if the entire shuffled card deck prints correctly
- Two points each for 7 separate test cases that create and print a single card value. Some of these cases will be error handling cases.
- Zero to 10 points based on the grader's evaluation of `MyCardDeck`'s software clarity and design quality