**Домашнее задание: Java**

1. Выполните упражнения на <https://www.w3schools.com/java/exercise.asp>.

При возникновении затруднений ознакомьтесь с туториалом, ссылка на который приведена в конце каждой секции.

Приведите скриншот, где можно видеть число выполненных упражнений.

В качестве решения заданий 2-5 требуются исходные коды описанных классов и демонстрационных программ (достаточно текстового вывода на консоль). Можно использовать Android Studio, IntelliJ Idea или любую другую IDE.

1. Реализуйте класс Card, представляющий игральную карту, характеризующуюся значениями rank (достоинство) и suit (масть - diamonds/clubs/hearts/spades). В простейшем варианте, оба этих значения могут быть целочисленными.

Класс Card нужно реализовать в нескольких вариантах:

* 1. С конструктором по умолчанию и двумя изменяемыми полями.
  2. С установкой начальных значений в конструкторе, закрытыми полями и методами чтения значений. Проверяйте значение suit – в случае ошибки выводите на экран соответствующее сообщение.
  3. Модификацию варианта *b*, с дополнительным конструктором, принимающим только suit. В качестве rank нужно устанавливать maxRank + 1, где maxRank – максимальное известное значение rank для заданного suit (может потребоваться одно или более статических полей).

*К примеру, после создания карт масти diamonds с rank =* ***4*** *и rank = 2, maxRank будет равен* ***4****. При вызове конструктора с указанием только масти diamonds, будет создана карта с rank = 5.*

В программе создайте несколько объектов класса версии *с,* используя как основной, так и дополнительный конструктор. Выведите на экран значения rank, чтобы убедиться в их корректности.

1. Модифицируйте класс версии *b* из предыдущего задания.

Реализуйте методы toString (с именами мастей и названиями достоинств – Jack, Queen, и т.д.), equals и hashCode.

Добавьте также методы:

* Проверяющий, что карта принадлежит к стандартной колоде (54 карты). В качестве джокеров можно принять карты любых двух мастей со значением большим, чем у туза.
* Проверяющий, является ли текущая карта сильнее другой (масть одинакова, значение больше).
* Сравнивающий текущую карту с другой. Предполагается, что для мастей clubs < diamonds < spades < hearts. Нужно вернуть значение меньше нуля если текущая меньше, 0 если они равны, и значение больше нуля, если текущая больше. Добавьте аналогичный статический метод, сравнивающий две карты.

*Примечание: при выполнении заданий 2-3 нужно задействовать массивы, если позволяет реализация. Смысл именно в знакомстве с этой конструкцией языка.*

1. Реализуйте класс Deck, представляющий стандартную колоду карт. Для хранения карт (из предыдущего задания) используйте ArrayList. Сделайте конструктор закрытым, а для создания экземпляров оформите статические методы для создания колоды с джокерами, либо без них.

При реализации методов по необходимости используйте средства из стандартной библиотеки. Реализация вручную, к примеру, перемешивания карт, ошибкой считаться не будет.

Добавьте методы

* Статический метод, возвращающий случайную карту из стандартной колоды.
* Для перемешивания карт.
* Для сортировки карт (реализуйте интерфейс Comparable<Card> в Card, используя добавленный ранее метод сравнения).
* Для раздачи карты (извлечения из колоды).
* Для возврата карты в колоду (убедитесь, что карта отсутствует в колоде).
* Для проверки на пустоту.
* Реализацию toString.

Напишите программу, в которой создаётся колода, раздаются несколько карт, какие-то из них возвращаются, некоторые пробуем вернуть дважды. Выведите содержимое колоды.

Перемешайте, и выведите содержимое, затем отсортируйте и снова выведите.

В цикле извлекайте карты, пока колода не окажется пустой. Выведите содержимое на экран.

1. В этом задании нужно разработать средства для представления иерархической структуры некой файловой системы.

Оформите интерфейс FileSystemNode с методами

1. Возвращающим родительский элемент типа FileSystemNode.
2. Возвращающим имя.
3. Возвращающим путь до элемента в условной файловой системе (формат произвольный).

Добавьте абстрактный класс AbstractFileSystemNode, в котором реализуйте перечисленные методы. Запретите переопределение 1-го и 3-го методов.

Реализуйте класс Folder, принимающий в конструкторе переменное число аргументов типа FileSystemNode (дочерних элементов). Не забудьте изменить родителя у дочерних элементов. Добавьте метод для получения списка дочерних элементов.

Реализуйте класс File. Добавьте метод, возвращающий расширение файла.

В тестовой программе создайте корневую папку, добавьте пару подпапок с несколькими файлами. Отобразите путь до какого-либо файла. Отобразите расширение N-го дочернего файла какой-либо папки.