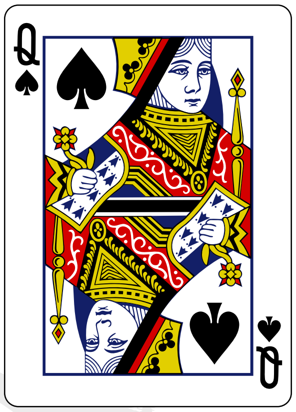
Lab 2: Playing Card Object



Create a functional `Card` object, that represents a playing card.

It is important that this is a full, professional `Card` object, like you might see in a real program. It should be immutable (meaning that every nonstatic field should be final). This Card also needs to implement the `Comparable<Card>` interface, in order to make it properly sortable. Every public method and field requires a Javadoc-style comment.

The Card should contain public static final variables for each suit and for each named card, that holds the proper value (e.g. `ACE` is 1, `KING` is 13). `CLUBS`, `DIAMONDS`, `SPADES`, and `HEARTS` should be 0, 1, 2, and 3 respectively.

The Card must contain each of the following methods. Those that override a built-in function should be preceded by an `@Override` directive.

- `Card(int rank, int suit, BufferedImage image)`, a constructor which takes a rank and a suit for the new card, as well as an image. If the image is null, it uses the pre-defined default image. It should throw an IllegalArgumentException if the rank and suit are not in the correct range.
- `Card(int rank, int suit)`, a constructor which takes a rank and a suit for the new card, and uses the default image. This constructor should call the 3-argument version.
- `Card()`, another constructor which produces a random `Card`, using the correct default image. This constructor should call the 3-argument version.
- `void draw(Graphics2D pen, int x, int y)`, which draws this `Card`'s image to the given x and y location.
- `boolean equals(Object object)`, which returns true iff (if and only if) this `Card` has the same rank and suit as the other.
- `boolean hasSameRank(Card other)`, which returns true iff the two `Cards` have the same rank.
- `boolean hasSameSuit(Card other)`, which returns true iff the two `Cards` have the same suit.
- `boolean hasGreaterRank(Card other)`, which returns true iff this `Card` outranks the other.
- `boolean hasLesserRank(Card other)`, which returns true iff the other `Card` outranks this `Card`.
- `boolean isFaceCard()`, which true iff the `Card` is a jack, queen, or king.
- `boolean isRed()`, which returns true iff the `Card` is hearts or diamonds.

- `boolean isBlack()`, which returns true iff the `Card` is spades or clubs.
- `String toString()`, which returns a human-readable String representing the `Card`.
- accessors (getters) for rank and suit. (Setters are not needed.)

In addition, you need this public static method:

- `setDefaultImages(BufferedImage[][] defaultImages)`, which takes a 2D array of `BufferedImage`s (like that returned by the appropriate version of `loadImageAsTiles()` in the `GraphicsWindow` class), and assigns the default images of the Cards to be from this array.

You must also create a `main()` method for testing, in order to unit-test your `Card`. It should perform the following steps:

1. Make a new `GraphicsWindow` large enough for 6 cards (plus some spacing around them).
2. Load the image file.
3. Make 6 new `Card` objects and draw them in this order so they don't touch each other: the Queen of Spades, the 6 of Clubs, the Ace of Diamonds, another 6 of Clubs, the 6 of Spades, and a 6th random Card.
4. Call the `GraphicsWindow`'s `finalize()` method, to make sure all the things you drew are properly displayed.
5. Use these `Card`s to test all of the above methods, printing out the test being performed and its result (e.g. "Queen of Spades is face card? true"). Each boolean method should be tested both in a situation where it returns `true`, and where it returns `false`.

You may also implement as many private methods as you wish, if you feel it will help.

Finally, try not to hard-code any values! Use static final variables to hold things like a card's size, the window's size, the image file name, and the number of ranks/suits, so that these things can be changed easily.

Your file should be called `Card.java`.