Small Assignment #7
Due: Wednesday, 4/9/2025 by 11:59 PM

Submission. This assignment needs to be submitted on D2L. Instructions are given at the end. Make sure you follow them very carefully because an incorrect submission will result in a 0 on the assignment.

Instructions.
You are given a folder of code. You need to implement the following things and then submit the code according to the instructions below.

CardStack.java

- A `CardStack` is a stack of cards, so it should support adding a card to the top of the pile (which is already implemented) and drawing a card from the top of the pile. Make `CardStack` implement the `Drawable` interface (which is provided). A card should be drawn from the top of the stack. Notice that the specification for draw in `Drawable.java` uses a precondition to note that the stack should not be empty, so you do not have to do anything additional to handle that. That's Design by Contract.
- Make `CardStack` iterable as well using the `Iterable` and `Iterator` interfaces as shown in class. The cards should be iterated through from top to bottom.

Card.java & Deck.java

- Implement (or use your IDE to generate) appropriate overrides for the `equals` and `hashCode` methods of `Card.java`. (In Eclipse, you can do this by going to Source → Generate hashCode() and equals() methods.)
- Make `Deck` implement `Drawable`. The draw method should remove and return the first card in the deck.
- Currently, Cards can only be ordered in one way. Change what needs to be changed in order to allow Cards to be sorted in two ways–(1) by rank and then suit & (2) by suit and then rank. Use the STRATEGY design pattern to make `Deck` sortable in two ways (1) by `Rank` first and (2) by `Suit` first.

Domino.java & DominoSet.java

- Implement (or use your IDE to generate) appropriate overrides for the `equals` and `hashCode` methods of `Domino.java`. (In Eclipse, you can do this by going to Source → Generate hashCode() and equals() methods.)
- Implement (or use your IDE to generate) an appropriate toString method for `Domino.java`.
- Make appropriate changes to `Domino` and `DominoSet` so that you can easily sort the Dominos by the sums of their two sides in ascending order.
- Make `DominoSet Drawable`. Drawing from the set should remove and return the first one in the list.
- Make `DominoSet Iterable`.

DrawableSet.java

- This class makes it easy to manage multiple collections of cards at once. The card collections could be anything that implements `Drawable<Card>` (e.g. `Deck`, `CardStack`, `DrawableSet`).
- Make `DrawableSet` implement `Drawable<Card>`.
    - The `isEmpty` method should return true if and only if ALL the collections of cards are empty.
    - The `draw` method should go through each set and call `draw` on the first non-empty one.
- The `addSet` method in `DrawableSet` has an escaping reference. We could fix this by making a copy except that `Drawable` is a general type, so we don't know how to copy it. The solution to this is simple – add to the `Drawable` interface to make sure we know how to copy anything that is `Drawable`.
- Add the following abstract method to the `Drawable` interface: `Drawable copy();` – this should return a copy of the item.
- Fix all the classes that implement `Drawable` by implementing the `copy` method. Note that in each case, you can return the specific object, not the general `Drawable` type. For example, the method signature for the copy method in `Deck` should be: `Deck copy()`
- Fix the escaping reference in `DrawableSet` using the `copy` method.

Main.java

- Do not edit this class.
- Make sure your code compiles and runs with the original `Main.java` class.
- Make sure you do not have any problematic escaping references.
- Note that this means that you may have to implement some additional methods in the classes. Add those as necessary, but do not introduce any problematic escaping references.
- Make sure your code works with the provided junit tests.

Submission.

- Put all the `.java` files into an executable jar file, where the executable class is `Main.java`.
- Make sure you can run it in the terminal using the jar file.
- Put all the `.java` files and the jar file in the same folder. Zip it up. Submit it to D2L.