Lab 8 Comments

- keep in mind the difference between creating a *variable* and creating a new *object*
  - while you often create a new variable when you create a new object, you also often create new variables without creating new objects
    - is this variable going to hold something new, or something that already exists?
- declare variables as locally as possible

```
Card card;     // declaration only – don't need a new Card!
for ( ; deck.cardsLeft() > 0 ; ) {
  card = deck.dealCard();
  …
}
```

```
for ( ; deck.cardsLeft() > 0 ; ) {
  Card card = deck.dealCard();
  …
}
```

`card` is only needed temporarily, to hold the card being handled in this iteration of the loop

Lab 8 Comments

- use of `Card`

Field Summary

Fields

Modifier and Type	Field	Description
static int	ACE	
static int	CLUBS	
static int	DIAMONDS	
static int	HEARTS	
static int	JACK	
static int	JOKER	
static int	KING	
static int	QUEEN	
static int	SPADES	

  - use `Card.CLUBS`, `Card.DIAMONDS`, etc – better than working with the suit as a `String` (which also requires knowing exactly what the `String` is for each suit)

Lab 8 Comments

- use of `Card`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
int	getSuit()	Returns the suit of this card.
java.lang.String	getSuitAsString()	Returns a String representation of the card's suit.
int	getValue()	Returns the value of this card.
java.lang.String	getValueAsString()	Returns a String representation of the card's value.
java.lang.String	toString()	Returns a string representation of this card, including both its suit and its value (except that for a Joker with value 1, the return value is just "Joker").

  - prefer `getSuit()` (and working with `Card.HEARTS`, etc) to `getSuitAsString()` for comparing suits
    - you don't need to know what the exact strings are
    - because suits are `int`s, can use == instead of `equals` to compare
  - `toString()` means that `System.out.println(card)` is nice
    - no need to call `toString()` explicitly – the system will call it when a `Card` is used in a `String` context
    - no need to retrieve the suit and value separately to print a `Card`

Lab 8 Comments

- use an `if` structure that reflects the nature of the alternatives

```
int suit = card.getSuit();
if ( suit == Card.HEARTS ) { … }
if ( suit == Card.CLUBS ) { … }
if ( suit == Card.SPADES ) { … }
if ( suit == Card.DIAMONDS ) { … }
```

the alternatives are mutually exclusive (`suit` can't have more than one value at a time) so both of these work correctly, but the first is structured as a series of independent choices – in principle, more than one of the alternatives could happen – while in the second it is clear that at most one alternative will happen

```
int suit = card.getSuit();
if ( suit == Card.HEARTS ) { … }
else if ( suit == Card.CLUBS ) { … }
else if ( suit == Card.SPADES ) { … }
else if ( suit == Card.DIAMONDS ) { … }
```

having your code structure match the nature of the situation makes it clearer and helps prevent bugs

Lab 8 Comments

- animation
  - only things that change (or potentially change) should go into "update animation variables" in `drawFrame`
  - setting the speed of the banners is initialization, and goes in the "initialize animation variables" section