15.1

/\*

\* Merges two sorted decks into a new sorted deck. Now an \* object method

\*/

**public** Deck merge(Deck d1) {

// create the new deck

Deck result = **new** Deck (cards.length + d1.cards.length);

**int** choice; // records the winner (1 means d1, 2 means d2)

**int** i = 0; // traverses the first input deck

**int** j = 0; // traverses the second input deck

// k traverses the new (merged) deck

**for** (**int** k = 0; k < result.cards.length; k++) {

choice = 1;

// if d1 is empty, d2 wins; if d2 is empty, d1 wins; otherwise,

// compare the two cards

**if** (i == cards.length)

choice = 2;

**else** **if** (j == d1.cards.length)

choice = 1;

**else** **if** (cards[i].compareTo(d1.cards[j]) > 0)

choice = 2;

// make the new deck refer to the winner card

**if** (choice == 1) {

result.cards[k] = cards[i]; i++;

} **else** {

result.cards[k] = d1.cards[j]; j++;

}

}

**return** result;

}

Version I prefer: I prefer when merge is a class method. As an object method, merge becomes more confusing and does not appear to be any more efficient than as a class method.

15.2

**public** **double** abs() {

**return** Math.*sqrt*(real \*real + imag \* imag);

}

15.3

**public** **static** **boolean** equals(Complex a, Complex b) {

**return**(a.real == b.real && a.imag ==b.imag);

}

15.4

Pros and cons:

For this example, I think it definitely makes more sense to keep these methods as object methods. When class methods are used, the code becomes confusing and hard to follow. When they are object methods, you can easily see which object is calling the method and what it is doing. In this case, object methods feel “more natural”. Class methods make more sense for computations such as comparing different objects. Class methods also make more sense for computations that don’t require the values or data of an object.

My 15.4 Rational.java is in the zip.

15.5 is in the zip as CardSoln3.java