

Prelude: Designing Classes

1. Consider the interface `NameInterface` defined in Segment P.13. We provided comments for only two of the methods. Write comments in javadoc style for each of the other methods.

```
/** Sets the first and last names.
 * @param firstName A string that is the desired first name.
 * @param lastName A string that is the desired last name. */
public void setName(String firstName, String lastName);

/** Gets the full name.
 * @return A string containing the first and last names. */
public String getName();

/** Sets the first name.
 * @param firstName A string that is the desired first name. */
public void setFirst(String firstName);

/** Gets the first name.
 * @return A string containing the first name. */
public String getFirst();

/** Sets the last name.
 * @param lastName A string that is the desired last name. */
public void setLast(String lastName);

/** Gets the last name.
 * @return A string containing the last name. */
public String getLast();

/** Changes the last name of the given Name object to the last name of this Name object.
 * @param aName A given Name object whose last name is to be changed. */
public void giveLastNameTo(NameInterface aName);

/** Gets the full name.
 * @return A string containing the first and last names. */
public String toString();
```

2. Consider the interface `Circular` and the class `Circle`, as given in Segment P.15.
 - a. Is the client or the method `setRadius` responsible for ensuring that the circle's radius is positive?
 - b. Write a precondition and a postcondition for the method `setRadius`.
 - c. Write comments for the method `setRadius` in a style suitable for javadoc.
 - d. Revise the method `setRadius` and its precondition and postcondition to change the responsibility mentioned in your answer to Part a.

```
a. The client is responsible for guaranteeing that the argument to the setRadius method is positive.
b. Precondition: newRadius >= 0. Postcondition: The radius has been set to newRadius.
c. /** Sets the radius.
 * @param newRadius A non-negative real number. */
d. Precondition: newRadius is the radius. Postcondition: The radius has been set to newRadius if newRadius
   >= 0.
   /** Sets the radius.
    * @param newRadius A real number.
    * @throws ArithmeticException if newRadius < 0. */
public void setRadius(double newRadius) throws ArithmeticException
{
    if (newRadius < 0)
        throw new ArithmeticException("Radius was negative");
    else
        radius = newRadius;
} // end setRadius
```

3. Write a CRC card and a class diagram for a proposed class called **Counter**. An object of this class will be used to count things, so it will record a count that is a nonnegative whole number. Include methods to set the counter to a given integer, to increase the count by 1, and to decrease the count by 1. Also include a method that returns the current count as an integer, a method **toString** that returns the current count as a string suitable for display on the screen, and a method that tests whether the current count is zero.

Counter
Responsibilities Set the counter to a value Add 1 to the counter Subtract 1 from the counter Get the value of the counter as an integer Get the value of the counter as a string Test whether the counter is zero Collaborations

Counter
-count: integer
+setCounter(theCount:integer): void +incrementCount(): void +decrementCount(): void +getCurrentCount(): integer +toString(): String +isZero(): boolean

4. Suppose you want to design software for a pharmacy. Give use cases for purchasing medicines and settling the bill. Identify a list of possible classes. Pick two of these classes, and write CRC cards for them.

System: Purchases

Use case: Purchase the Medicine

Actor: Pharmacist

Steps:

1. The pharmacist initiates a new purchase.
2. The pharmacist enters a purchase number.
3. The pharmacist chooses the medicine and adds it to the purchase.
 - a. If there are more medicines, return to step 3.
4. The purchase is forwarded to the billing section.

System: Purchases

Use case: Settle Bill

Actor: Cashier

Steps:

1. The cashier enters the purchase ID.
2. The system displays the total.
3. The customer makes a payment to the cashier.
4. The system computes any change due.
5. The cashier gives the customer a receipt.

Possible classes for this system are Pharmacy, Pharmacist, Cashier, ListOfMedicines, Medicine, Purchase, PurchaseMedicine, and Payment.