CS 142: Program design and methodology
Spring Term, 2026

Practice Problems on Testing

Complete the following problems.

1. Consider a `Refrigerator` class with the methods below.

   - A constructor that takes the refrigerator's current temperature and its desired temperature. For this problem, temperatures will always be integers.
   - A method `chill`, which lowers the current temperature by 1 degree if it is currently above the desired temperature. The method returns the refrigerator's current temperature.
   - A method `open`, which causes the current temperature to move 5 degrees toward 70. (The temperature increases if it is below 70 and decreases if it is above 70.) The amount of change will be less than 5 if changing this much would cause the temperature to pass 70. (For example, it only increases from 69 to 70 since a greater increase would make the temperature above 70.)
   - A method `adjustTemp`, which takes a new desired temperature for the refrigerator. The desired temperature is set to this value unless the new temperature is below 45 or above 65, in which case the desired temperature is unchanged.

   Based on this description, write JUnit test cases for `Refrigerator` using the setUp method I've provided.

```
public class TestRefrigerator {
  private Refrigerator r;

  @Before
  public void setUp() {
    r = new Refrigerator(65, 50);
  }
```

2. The `VacuumCleaner` class models a rechargable vacuum cleaner with a fixed capacity for dirt. The class has the following methods:

   - A constructor that takes an integer argument and creates a `VacuumCleaner` that is fully charged and has a dirt capacity equal to the argument. The argument is assumed to be positive.
   - A method `use` that takes an integer which is the amount of dirt that is being cleaned. This amount is assumed to be positive. If the vacuum cleaner is charged, it cleans up to the minimum of the argument value and its remaining capacity (i.e. its overall capacity minus the amount of dirt it already contains). This amount of dirt is added to its current contents and returned by the method. The vacuum also becomes uncharged. If the vacuum cleaner is uncharged when the method is called, it throws an `UnsupportedOperationException` exception. (We should probably create our own, more appropriate, exception, but that's beyond the scope of this assignment.)
   - A method `charge` that takes no arguments and makes the vacuum fully charged.
   - A method `empty` that takes no arguments and causes all the dirt in the vacuum cleaner to be removed. It returns the amount of dirt removed.

   Based on this description, write JUnit test cases for `VacuumCleaner` using the `setUp` method I've provided.

```
public class TestVacuumCleaner {
  private VacuumCleaner v;

  @Before
  public void setUp() {
    v = new VacuumCleaner(10);
  }
```