Homework 13

Shyam Sai Bethina

24 October 2021

```java
//1
/**
 * Returns the product of the digits of {@code n}.
 *
 * @param n
 *            {@code NaturalNumber} whose digits to multiply
 * @return the product of the digits of {@code n}
 * @clears n
 * @ensures productOfDigits1 = [product of the digits of n]
 */
private static NaturalNumber productOfDigits1(NaturalNumber n) {
    NaturalNumber product = new NaturalNumber2(1);
    int lastDigit = n.divideBy10();
    NaturalNumber temp = new NaturalNumber2(lastDigit);
    if(!n.isZero()) {
        temp.multiply(temp);
        productOfDigits1(n);
    }

    n.clear();
    return product;
}

/**
 * Returns the product of the digits of {@code n}.
 *
 * @param n
 *            {@code NaturalNumber} whose digits to multiply
 * @return the product of the digits of {@code n}
 * @ensures productOfDigits2 = [product of the digits of n]
 */
private static NaturalNumber productOfDigits2(NaturalNumber n) {
    NaturalNumber product = new NaturalNumber2(1);
    int lastDigit = n.divideBy10();
    NaturalNumber temp = new NaturalNumber2(lastDigit);
    if(!n.isZero()) {
        temp.multiply(temp);
        productOfDigits1(n);
        n.multiplyBy10(lastDigit);
    }
    return product;
}
```

```java
//2
/**
 * Reports the value of {@code n} as an {@code int}, when {@code n} is
 * small enough.
 *
 * @param n
 *            the given {@code NaturalNumber}
 * @return the value
 * @requires n <= Integer.MAX_VALUE
 * @ensures toInt = n
 */
private static int toInt(NaturalNumber n) {
    NaturalNumber max = new NaturalNumber2(Integer.MAX_VALUE);
    if(n.compareTo(max)>0) {
        n.divideBy10();
        toInt(n);
    }
    int returnMax = n.toInt();
    return returnMax;
}
```

```java
//3
/**
 * Reports whether the given tag appears in the given {@code XMLTree}.
 *
 * @param xml
 *            the {@code XMLTree}
 * @param tag
 *            the tag name
 * @return true if the given tag appears in the given {@code XMLTree},
 *         false otherwise
 * @ensures <pre>
 * findTag =
 *    [true if the given tag appears in the given {@code XMLTree}, false otherwise]
 * </pre>
 */
private static boolean findTag(XMLTree xml, String tag) {
    boolean appears = false;
    if(!appears) {
        if(xml.isTag()) {
            appears = xml.label().equals(tag);
            for(int i=0;i<xml.numberOfChildren() && !appears;i++) {
                findTag(xml.child(i),tag);
            }
        }

    }
    return appears;
}
```

4) i) A software design framework that splits up issues between clients and implementors. Uses preconditions and postconditions in order to clearly communicate issues and purposes of each software component

ii) characterizes the responsibility of the program that calls the code

iii) characterizes the responsibility of the program that implements that method or software component

iv) a way to check if each line of code is executed correctly, and the correct output is displayed. If not, then change the line of code that is making the program faulty

v) Using breakpoints, debugging allows the person writing code to look at the values of the variable and control the flow of code execution

vi) parameter mode indicates how a possible way that a method might change the value of the argument

vii) clears mode clears the incoming argument to its original value

viii) this mode replaces this with the incoming value

ix) restores the incoming argument to its original value before the method call

x) this mode changes this based on the incoming argument value

xi) an type where the initialized value cannot be changed

xii) types that include int, double, Boolean, char, bytes

xiii) Reference types are all the other types that are not primitive types, like Strings and there can be as many user-created types as you want

xiv) an object is something defined by classes and interfaces

xv) When more than one variable references the same object value

xvi) the type of interface the variable will use, on the left side of the assignment operator

xvii) object type is the class the variable will take on, on the right side of the assignment operator

xviii) when a class implements an interface, then the class has code for the method bodies as described in the interface

xix) extends allows the class to inherit all the method of the class it is extending to, and can add method contracts to its own class

xx) when the class that is extending changes a method that was already defined in the class that it extends to

xxi) when an interface extends another interface, it becomes a sub interface of the interface it extends to

xxii) if a class a extends class b, class b is the parent class as it is higher in the hierarchy than class a

xxiii) polymorphism is when the same method changes on the type of object the receiver is

xxiv) recursion is when a method calls itself within the method body

5) Each element in the array is an alias to the same value, so they will all point to the value 5 when the program ends, because they all point to the count variable. I would change this by initializing the count variable inside the for loop and change new NaturalNumber2(1) to new NaturalNumber2(i+1) and remove the count.increment() statement.

6) NaturalNumber has methods bodies from the Standard interface and has its own method bodies on top of the method bodies from the Standard interface, meaning that NaturalNumber extends the Standard interface

7) Since C4 extends C3, it has methods bodies from C3. Since C3 implements I2, C3 method bodies has code that are described in the I2 interfaces, meaning that C4 also has method bodies that has code working the way it is described in the I2 interface. Since I2 has method bodies from I2 and its own method bodies, I2 extends I1, and C3 has code for all of those method bodies from both I2 and I1, so C3 implements I1.