Solutions for task-13

By Sara Rydell

2023-01-19

Exercise 13.1.1 - Inheritance

a. Determining which of the following assignments are legal

These assignments are legal since the objects created are in all these cases sub classes to the super or parent class in which they are defined:
Person p1 = new Student();
Person p2 = new PhDStudent();
Student s1 = new PhDStudent();

These assignments are illegal since we have the reversed situation where a super class object is defined from the sub classes, which isn't possible:
PhDStudent phd1 = new Student();
Teacher t1 = new Person();

b. Determining which of the following assignments are legal

These assignments are legal because a sub class can be partially defined through a parent class since booth of them have the overlapping attributes:
p1 = s1;
t1 = s1;
s1 = phd1;

These assignments are illegal because a super or parent class can't be defined through a sub class since the sub classes will have additional attributes which can not be mapped to the parent:
s1 = p1;
s1 = p2;
phd1 = s1;

Exercise 13.1.2 (src - use newsfeed)

This solution can be found in the EventPost.java file.

Exercise 13.1.3

If I remove the "extends Post" from the class defenition of EventPost and then call NewsFeed.addPost I will get several errors. This is because this statement makes the class EventPost a sub class to the super class Posts. That's why we will get one error about incompatible types since the classes Post and EventPost does not belong to the same class type. This also does so that we can't use the super() instruction to connect

attributes of Post to EventPost. We also an error from the addPost() method since the method we creted in the EventPost class can't be called in the Post class.

If I were to remove super() from the constructor of EventPost and then call NewsFeed.show() then the constructor Post in class Post can't be applied to the given type. This can be because the program in this case will use the addPost() method can't call the Post in EventPosts since it's not part of the class Post. Then it might assume that we are trying to use the incorrect input for the kind of post we are trying to make.

If I were to remove super.display() from the display methods in EventPost and then call NewsFeed.show, then the EventPost content will be displayed although without the information specified by this method being the username/author of the post, time since posting and number of likes.

When we have two classes with an inheritance relationship and they have a method with the same signature that is called method overriding.

Exercise 13.2 - Induction Warmup

Proofs by induction that the following statements are true for all natural numbers.

$$\sum_{j=1}^{n}(2j-1) = n^2 \tag{1}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

Proof statement 1:

Base step – Show that eq. (1) holds when $n = 1$.

$$VL = 2 * 1 - 1 = 2 - 1 = 1 = 1^2 = HL \tag{3}$$

So booth sides are equal and (1) is true for $n = 1$.

Inductive step – First, let p belong to the natural numbers and assume that our statement is true for all $p = n$.

$$\sum_{j=1}^{p}(2j-1) = p^2 \tag{4}$$

Second, prove that the statement also holds true for $p + 1$.

$$\sum_{j=1}^{p+1}(2j-1) = \sum_{j=1}^{p}(2j-1) + (2(p+1) - 1 \tag{5}$$

By our induction hypothesis we get the following:

$$VL = p^2 + (2(p+1) - 1) = p^2 + 2p + 2 - 1 = p^2 + 2p + 1 = (p+1)^2 = HL \tag{6}$$

Conclusion – And we are done! We have proven by induction that the statement in eq. (1) is indeed true.

Proof statement 2:

Base step – Show that eq. (2) holds when $n = 1$.

$$HL = \frac{1(1+1)(2*1+1)}{6} = \frac{2*3}{6} = 1 = 1^2 = VL \tag{7}$$

So booth sides are equal and (2) is true for $n = 1$.

Inductive step – First, let p belong to the natural numbers and assume that our statement is true for all $p = n$.

$$\sum_{i=1}^{p} i^2 = \frac{p(p+1)(2p+1)}{6} \tag{8}$$

Second, prove that the statement also holds true for $p + 1$.

$$VL = \sum_{i=1}^{p+1} i^2 = \sum_{i=1}^{p} i^2 + (p+1)^2 = \sum_{i=1}^{p} i^2 + (p+1)^2 = \sum_{i=1}^{p} i^2 + p^2 + 2p + 1 \tag{9}$$

By our induction hypothesis we get the following:

$$VL = \frac{p(p+1)(2p+1)}{6} + p^2 + 2p + 1 = \frac{(p^2+p)(2p+1)}{6} + \frac{6p^2+12p+6}{6} = \tag{10}$$

$$= \frac{2p^3 + p^2 + 2p^2 + p}{6} + \frac{6p^2+12p+6}{6} = \frac{2p^3+3p^2+13p+6}{6} \tag{11}$$

$$HL = \frac{(p+1)(p+1+1)(2(p+1)+1)}{6} = \frac{(p+1)(p+2)(2p+3)}{6} = \frac{(p^2+3p+2)(2p+3)}{6} = \tag{12}$$

$$\frac{2p^3+3p^2+6p^2+9p+4p+6}{6} = \frac{2p^3+3p^2+13p+6}{6} \tag{13}$$

Thus we have proven that $VL = HL$ is true.

Conclusion – And we are done! We have proven by induction that the statement in eq. (2) is indeed true since booth sides of the equations are equal.

Exercise 13.3.1

Invariant: $res = x^i$

This is because x will res will be multiplied with x equally many times as the loop with the counter i is called.

Exercise 13.3.2

When running the function we get the following:
Initializing the constant res will be done 1 time. The for loop and incrementation of the constant i will in total be done n times. Then for each loop res will be multiplied with x one time, giving us n times. Finally res will be returned one time.

Time complexity for expIterative: $t(n) = 1 + n + n + 1 = 2n + 2$

Big-O notation:
$f(n)$ is $O(g(n))$ if there exists constants c and n_0 so that: $o \leq f(n) \leq c * g(n)$ for all $n \geq n_0$

Let $f(n) = 2n + 2$ and $g(n) = n$ where $c = 3$ and $n_0 = 2$

This means that the function is O(n) in Big-O notation.

The basic operation is the res *= x; operation and in regards to this we can directly get that the complexity is g(n) = n and thus the Big-O notation result O(n).

Exercise 13.3.3

Here is a proof by induction for **expRecursive** that proves its correctness.
In this case we already can assume the correctness of expIterative for all $n \geq 0$

The algorithm:

$$expRecursive(x, n) = expRecursive(x, \frac{n}{2}) * expRecursive(x, \frac{n+1}{2}) \tag{14}$$

Mathematical expression of algorithm:

$$x^n = x^{\lfloor \frac{n}{2} \rfloor} * x^{\lfloor (\frac{n+1}{2}) \rfloor} \tag{15}$$

Proof by induction that the statement above is true for all positive integers $n > 4$.

Base step – Show that the statement holds for the base case for odd and even numbers.

Odd numbers – when n=5 we get the following:

$$x^{\lfloor \frac{5}{2} \rfloor} * x^{\lfloor (\frac{5+1}{2}) \rfloor} = x^{\lfloor \frac{5}{2} \rfloor} * x^{\lfloor (\frac{6}{2}) \rfloor} = x^2 * x^3 = x^5 \tag{16}$$

Even numbers – when n=6 we get the following:

$$x^{\lfloor \frac{6}{2} \rfloor} * x^{\lfloor (\frac{6+1}{2}) \rfloor} = x^{\lfloor \frac{6}{2} \rfloor} * x^{\lfloor (\frac{7}{2}) \rfloor} = x^3 * x^3 = x^6 \tag{17}$$

In booth base cases booth sides of the equation are equal, thus our expression is true for $n = 5$ and $n = 6$.

Inductive step – First, let p belong to the integers $p > 4$ and assume that our statement is true for all $p = n$.

$$x^p = x^{\lfloor \frac{p}{2} \rfloor} * x^{\lfloor (\frac{p+1}{2}) \rfloor} \tag{18}$$

Second, prove that the statement also holds true for $p + 1$.

$$x^{\lfloor \frac{p+1}{2} \rfloor} * x^{\lfloor (\frac{p+1+1}{2}) \rfloor} = x^{\lfloor \frac{p+1}{2} \rfloor} * x^{\lfloor (\frac{p+2}{2}) \rfloor} = x^{\lfloor \frac{p+1}{2} \rfloor} * x^{\lfloor (\frac{p}{2}+1) \rfloor} =$$
$$= x^{\lfloor \frac{p+1}{2} \rfloor} * x^{\lfloor (\frac{p}{2}) \rfloor} * x^{\lfloor 1 \rfloor} = *by induction hypothesis* = x^p * x = x^{p+1} \tag{19}$$

Conclusion – And we are done! We have proven by induction that the statement in eq. (6) is indeed true.

Exercise 13.3.4

The recursive algorithm:

$$expRecursive(n) = \begin{cases} expRecursive(n/2) * expRecursive((n+1)/2) & \text{if } n > 4 \\ expIterative(n) & \text{if } n \leq 4 \end{cases} \tag{20}$$

Algorithm's recurrence:

$$T(n) = \begin{cases} 2T(n/2) + 1 & \text{if } n > 4 \\ n & \text{if } n \leq 4 \end{cases} \tag{21}$$

Using the Master theorem:
From the recurrence we decided that $a = 2$ since two recursive branches will be executed in expRecursive, $b = 2$ since for each recursive call the problem size will approximately be halved, and $c = 0$ since the complexity when the algorithm finishes the recursive calls are the complexity if expIterative which is n^0.

Using the Master theorem we get the following:
$2 > 2^0 = 1$ which gives us expRecursive's time complexity of $O(n^{log_2(2)}) = O(n)$