

PLD Assignment 3

Ask

17. marts 2022

Indhold

1	A3.1	1
1.1	a)	1
1.2	b)	1
1.3	c)	1
1.4	d)	2
1.5	e)	2
2	A3.2	2
2.1	a)	2
2.2	b)	3
3	A3.3	3
3.1	a)	3
3.2	b)	3
4	A3.4	3
4.1	a)	3
4.2	b)	3
4.3	c)	4
4.4	d)	4
5	A3.5	4
6	A3.6	4

1 A3.1

1.1 a)

The compiler rejects the program with a compile error, since the method "bingoString()" isn't defined for the generic type T.

This is the error message I get when I try to compile the program

```
Bingo.java:9: error: cannot find symbol
    System.out.println(t.bingoString());
                        ^
symbol: method bingoString()
location: variable t of type T
where T is a type-variable:
  T extends Object declared in class Bingo
1 error
```

1.2 b)

The following code should compile. But throw a NullPointerException. Otherwise, since we don't really know what BingoString is supposed to do in this specific code snippet, a runtime error could be caused by a wrong cast.

```
1  abstract class myAbstractClass {
2      public abstract String bingoString();
3  }
4
5  class Bingo<T extends myAbstractClass> {
6
7      public void dingo(T t) {
8          System.out.println(t.bingoString());
9      }
10
11
12     public static void main(String[] args) {
13         Bingo<myAbstractClass> myObj = new Bingo<>();
14         myObj.dingo(null);
15     }
16 }
```

1.3 c)

Reusing the same piece of code, as in 1b only with a minor tweaks. This code will compile but do nothing.

```
1  abstract class myAbstractClass {
2      public abstract String bingoString();
3  }
4
5  class Bingo<T extends myAbstractClass> {
6
7      public void dingo(T t) {
8          System.out.println(t.bingoString());
9      }
10
11     public static void main(String[] args) {
12
13     }
14 }
```

1.4 d)

Since we assume that the code will compile, this should lead to a run-time error. since we can't set a subclass equal to the value of a superclass in java.

1.5 e)

2 A3.2

2.1 a)

Advantages

- Extensive control over the program and which parts of the program that should be able to use which modules. Can change the export statement in descendant classes.
- Security. Having the possibility to restrict certain parts of the program from using specific modules. (This looks very similar to public, private, protected as we see in Java. Only at a slightly more general level in Eiffel.)

Disadvantages

- Risk of making too many internal functions unusable to other parts of the program. And or making the readability of the different exports a hassle to understand.
-

2.2 b)

Advantages

- Since every program in Modula-2 is composed of modules, code can easily be reused.
- Encapsulation. This gives the programmer the option to restrict visibility of certain parts of the code e.g. subprograms or data structures to other parts of a program.
- The possibility to only import specific methods from a module.

Disadvantages

- Importing a whole module only to use a couple of methods, may take up file size.
-

3 A3.3

3.1 a)

Plop is a function, that takes an empty list and some value and insert that value into the list. In the "second" line plop is called with a list pattern and the argument w. And then insert U and V in the start of the list and then insert w at the end of the instantiated list.

3.2 b)

4 A3.4

4.1 a)

Root and mixed are two different functions. Root creates a tree that has a root at X with either zero, two or three children. Mixed is a function that checks whether the tree created by the root function is a mixed tree. And will possibly return a boolean value if the tree created by root is or is not indeed a mixed tree.

4.2 b)

```
Root("gorilla",
    Root("goat",
        Root("duck",
            Root("koala"), Root("manatee")
        )
    )
    Root("impala")
```

```
)  
  Root("horse"), Root("ostrich")  
)
```

4.3 c)

Creating a method called: leftmost

I have the following.

```
letmost(Root(E),E).  
leftmost(Root(_, T1, _), E) :- leftmost(T1, E).  
leftmost(Root(_, T1, _, _), E) :- leftmost(T1, E).
```

4.4 d)**5 A3.5****6 A3.6**