

Lab report 1

1.5

1. Explain why the actual runtime type of subAsBase is not taken into account when invoking aMethod(). Try the other invocations that are deactivated currently and explain the results of invoking aMethod() there

When the object subAsBase is instantiated and declared as an object of type ABaseClass, a class is created of type sub-class but the object is of type ABaseClass. This means the object will be recognized as ABaseClass but it will have methods from ASubClass. The aMethod is found multiple times in the classes and the program will select method depending on which have matching parameters.

2. Explain why the implementation in the visitor package does not work. In the VisitorTest class, there is a small but incorrect program that is intended to be used to illustrate the Visitor pattern. There are two errors: one that gives error messages and another that gives an incorrect result. The output of the Visitor should match one of the toString functions.

The problem is found in the accept-method of the class Minus. The `v.visit(leftOperand)` and `v.visit(rightOperand)` have operands declared as "AbstractExpression". When passing these through to Visitor the method `visit(AbstractExpression expr)` will be called which gives an error message.

Instead we replace the lines with:

```
leftOperand.accept(v);  
and  
rightOperand.accept(v);
```

This will instead call the method with the right object type.

The wrong result depends on that the program print the expression in the wrong order →

```
leftOperand.accept(v);  
v.visit(this);  
rightOperand.accept(v);
```

1.5.1/1.5.2

In order to add the functionality to compare a condition to different values we add an extra class called "CaseConditional" that extends CompoundExpression and override its methods. This class will be used every time an input starts with the word "case". To be able to recognize the word "case" the CompoundExpression class will be extended with an "else if"-statement. When the statement is entered an new "CaseConditional" will be created. The CaseConditional will take in the parameters: a condition and a list of expressions. In the evaluate method we loop through the list of expressions to find the right value to return.