Gremaud Lucien
Paunoiu Bruno

Serie 3

IN.2028
March 12, 2019

1 Theory

a) With inheritance we can create classes that have a lot of characteristics in common with a class that already exists. For example, we can have a class Point which makes objects with x and y coordinates that are unchangeable(private fields) and the a class MoveablePoint that inherits the private fields from Point but with a public method that can change them.

b) Because an object from a child class is also a an object from the parent class but not the other way around. This can be useful, if we look at the example from the previous exercice, we could declare a variable but not know yet if its gonna be a Point or a MoveablePoint, so we just have to declare it as Point. If we do know that the point is going to move, we can declare it as MoveablePoint and this way we can be sure that the variable can be moved.

c) Only one. If there were more parent classes they could have conflicting fields or methods. For example, we couldnt use the super keyword.

d) Because the first thing that our new constructor must do, is to construct an object, so the first thing it does is call the parent constructor.

e) The visibility must be less strict than the visibility of the parent method and the type returned must remain the same. It is recommended to put @override before overriding.

f) final field means the field cant be modified, final method means the method cant be overridden, final class means the class cant have child classes.

g) It means that the variable is always going to refer to the same object, but the object can actually be modified.

h) If we want to make sure that a non-primitive variable doesnt change the object it references in the method, we can simply make it final. This also helps for readability.

i) Polymorphism is the fact that a method can have different implementations, depending on the object that calls the method. For example in the Bike class, getDevelopment depends on gearCoefficients and in SportBike it also depends on ringCoeffs so the method is different depending on the objects class.

j) No. We should use protected instead.

k) It is less versatile than simply using polymorphism. For example if we use instanceof every time theres a special behavior, when a new child class is created, we must search every use of instanceof and make sure it works with the new child class.

l) It is used to make a copy of an object. But if the clone() method is badly implemented, the clone method could actually copy the reference instead of the instance.

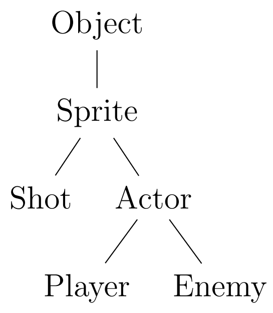
3 Debugging

a)
```
Error:(9, 35) java: constructor Car in class core.Car
  cannot be applied to given types;
  required: java.lang.String
  found: no arguments
  reason: actual and formal argument lists differ in length
```

c) Because the subclass ElectricCar do not override the method toString(), therefore it does not inblude the field ”enginePower”.

5 Project

5.1 Design & Implementation



5.2 Additional Lists

Shot and Enemy can be stored in SpriteList since they are both Sprites. The problem might be that Shot would not be handled correctly and would saturate the list. Also, if we want to apply a method to the entire list it might cause some conflicts.