Exercises:

1. Revisit the architecture of the *Car*-hierarchy you have created for the exercises of the last lecture.
   a) Add at least one abstract class or make an existing class abstract.
   b) Implement toString() and equals() for all types as far as it makes sense.
   c) Mind to update the class diagram.
2. Have a look at the method *Objects.equals(Object, Object)*. – What does this method do? For what should it be used?
3. What happens in this piece of code:

```
Object o = 42;
```

4. Have another look at the class *Class*.
   a) What is "reflection"?
   b) Write a program, that reflects the type *String*. It should print as many information about *String* to the console in a formatted way showing a profile of the type *String*.
5. Architect the types Shape, Triangle, Circle, Rectangle and Square by the usage of inheritance. Not every type needs to be implemented in its own class, if helpful more types can be introduced. Respect following guidelines:
   a) Stick to the SOLID principle.
   b) Each type should reside in dedicated file.
   c) All fields needs to be encapsulated.
   d) One of the types needs to be an abstract class.
   e) Following public methods should be implemented reasonably: *getPosition()*/*setPosition()*, *getA()*/*setA()*, *getB()*/*setB()*, *getC()*/*setC()*, *getD()*/*setD()* and *getRadius()*/*setRadius()*.
   f) The method *toString()* should be overridden to generate a *String* representing the data position, a, b, c, d and radius where reasonable.
   g) You should be able to draw the type hierarchy on the whiteboard (e.g. with a UML diagram).
   h) Prove the functionality of that types with some unit tests.

Remarks:

- As always.