

Exercise

Class diagrams



Prof. Dr. Bernhard Rumpe
Lehrstuhl für Software Engineering
RWTH Aachen

<http://www.se-rwth.de/>

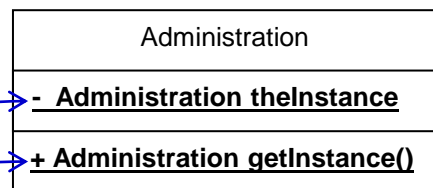
Exercise 2.1

- Where could you use the design pattern “Singleton“?
- Extend accordingly the class diagram.

Exercise 2.1



Stereotype «singleton» identifies the class as a singleton



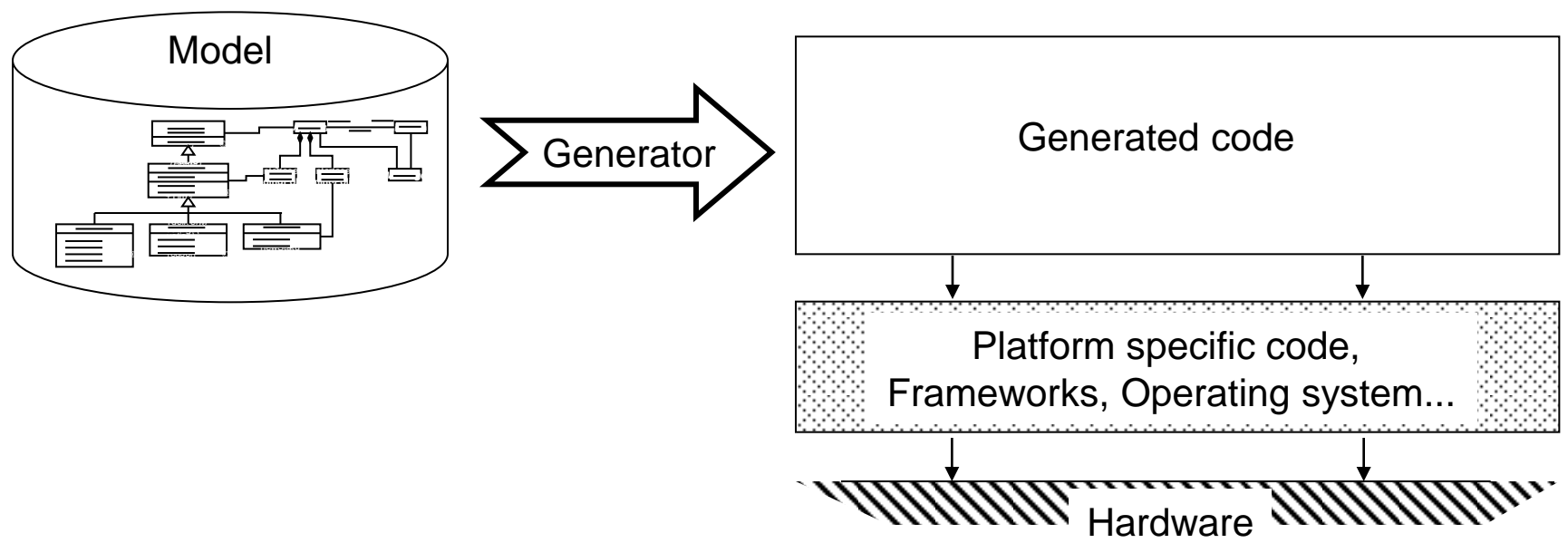
Elements of the Singleton pattern specified directly in the class



Additional UML-comment identifies the class as a singleton

Code Generation

- Principle: Mapping of the model into a programming language

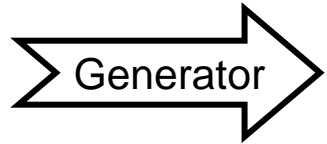


⇒ Even if there is no “automatic” generator available, it is possible to put the transformations of UML into code.

Code Generation from a Class

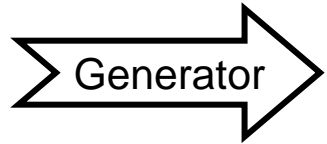
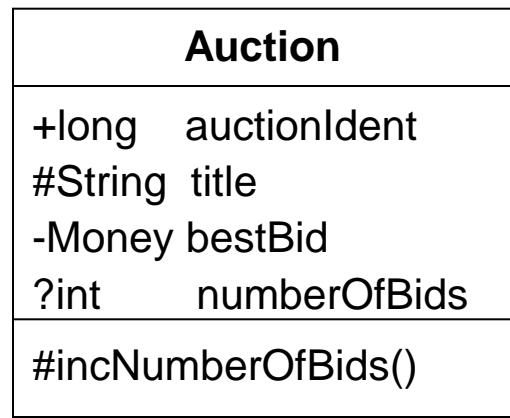
Auction	
+long	auctionIdent
#String	title
-Money	bestBid
?int	numberOfBids
#incNumberOfBids()	

CD



Suggestions?

Code Generation from a Class



```
class Auction {  
    public long    auctionIdent;  
    protected String title;  
    private Money  bestBid;  
    public int     numberOfBids;  
  
    protected void incNumberOfBids() { ... }  
}
```

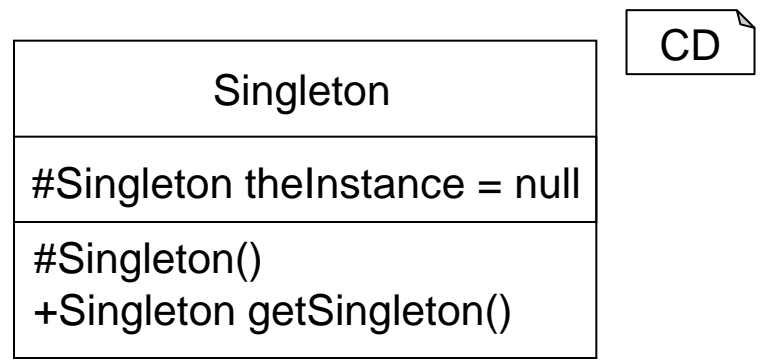


Exercise 2.2

- Suppose that the use of the design pattern “Singleton“ in a class is only marked with the use of the stereotype “Singleton“.
- Implement the code for this design pattern in a way it could have been automatically generated.

Solution 2.2

- Singleton-Pattern:
 1. Visibility of the constructors: private
 2. Static attribute as exclusive instance of the class
 3. Static method for the access to the static instance



- => Generation: Modification of the code and code generation
- If necessary class is marked as final
(prohibits creation of instances through inheritance)

Solution 2.2

```
public class Administration {  
    // CodeGeneration: Singleton Start  
    private static Administration theInstance = null;  
  
    public static Administration getInstance() {  
        if (theInstance == null) {  
            theInstance = new Administration();  
        }  
        return theInstance ;  
    }  
    // CodeGeneration: Singleton End  
  
    // CodeGeneration: Visibility changed to private  
    private Administration() {  
        ...  
    }  
    public void return(RentalContract contract) {  
        ...  
    }  
    ...  
}
```

JAVA

Solution 2.2

- Thread save variant:

```
public class Administration {  
    // CodeGeneration: Singleton Start  
    private static Administration theInstance = new Administration();  
  
    public static Administration getInstance() {  
        return theInstance ;  
    }  
    // CodeGeneration: Singleton End  
  
    // CodeGeneration: Visibility changed to private  
    private Administration() {  
        ...  
    }  
    public void return(RentalContract contract) {  
        ...  
    }  
    ...  
}
```

JAVA

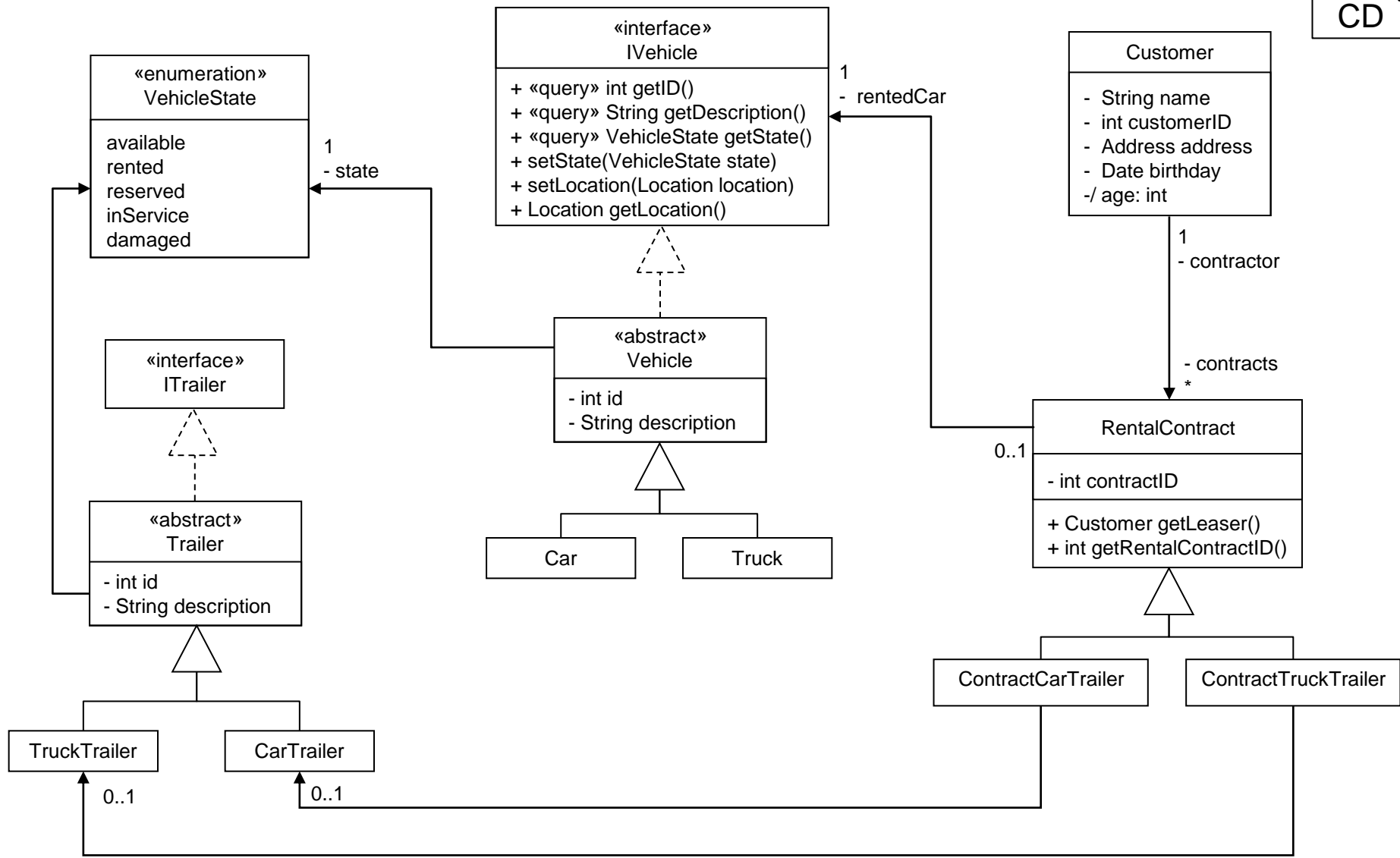
Exercise 2.3

After the car rental has extended its offer by trucks, it is possible to rent trailers for cars and trucks. But there are two different kinds of trailers: one type for every type of vehicle.

Change your class diagram in such a way that

- a trailer can only be rented with one vehicle and
- a truck trailer can only be rented in combination with a truck and a car trailer can only be rented with a car.

Solution 2.3



Questions?

