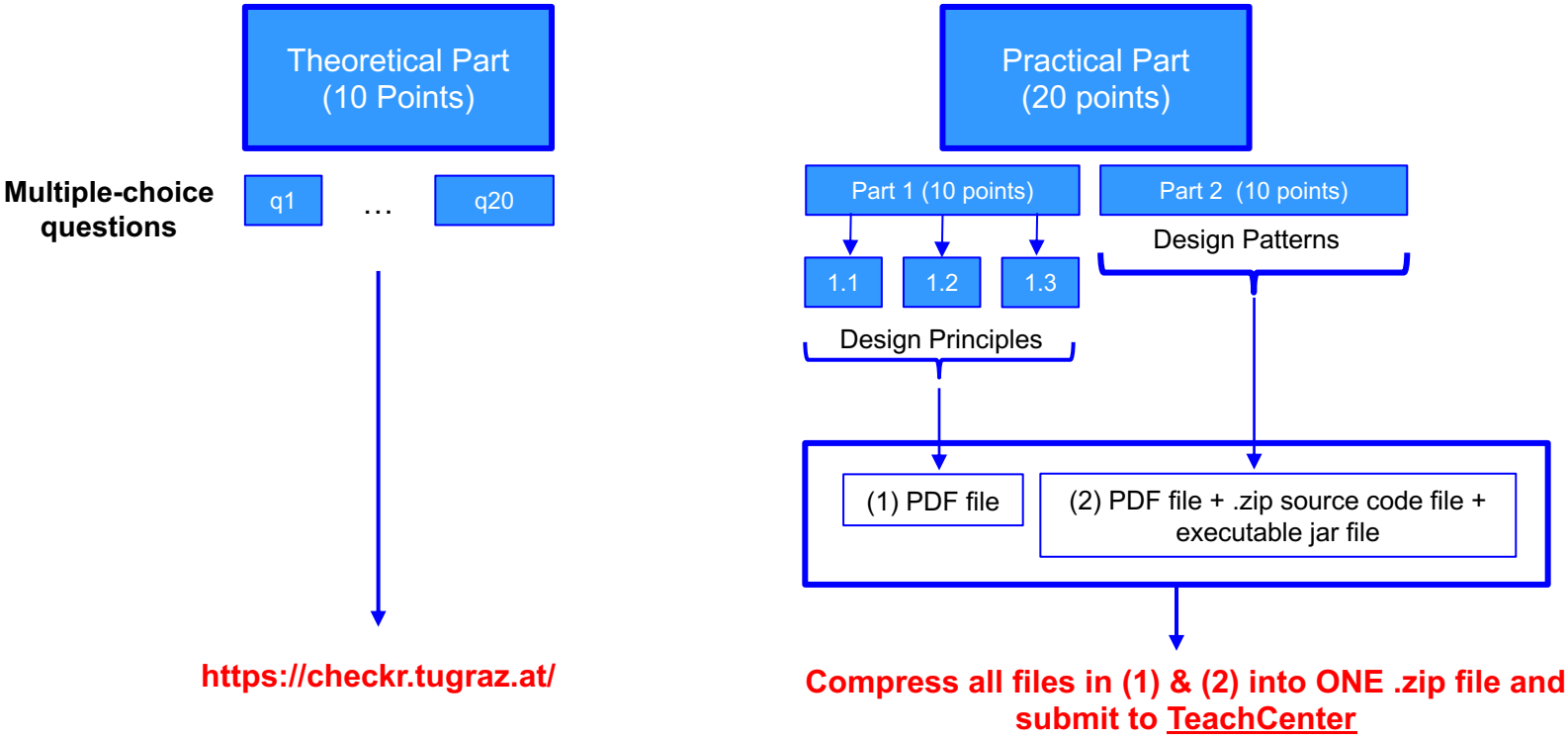
Objektorientierte Analyse & Design
- Übungsblatt 3 -
- Exercise 3 -

The structure of Exercise 3



Exercise 3 (Theoretical Part = 10 Points)

- Visit https://checkr.tugraz.at
- Register with your **TUGRAZ student mail address** (**important**)!
- At https://checkr.tugraz.at/, enter the key **wCRGuG**
- Provide your answers to the posed questions (several tries are possible!)
- The questions have to be answered individually
- If all questions have been answered correctly, this corresponds to **10 points**
- **No** submission is needed for the theoretical part!

Exercise 3 (Practical Part 1.1 = 3 Points)

Given a Java source code (as shown in the picture) specifying the *baggage allowance* in a flight based on the *passenger type* (e.g., *Economy, Flex, or Business*), please answer the following questions:

1. Which issue (in terms of the SOLID principles violation) could be triggered in the implementation? *(Name the principle and explain the issue)*
2. How to solve the issue? *(Provide a solution in the form of Java source code)*

Note: *Please create a PDF file and include all the answers to the questions.*

```java
enum PassengerType { Economy, Flex, Business };

class Passenger {
    private PassengerType passengerType;

    public PassengerType getPassengerType() {
        return passengerType;
    }

    public void setPassengerType(PassengerType passengerType) {
        this.passengerType = passengerType;
    }

    public double getBaggageAllowance() {
        if (this.passengerType == PassengerType.Economy)
            return 0;
        else if (this.passengerType == PassengerType.Flex)
            return 32;
        else
            return 64;
    }
}
```

Exercise 3 (Practical Part 1.2 = 3.5 Points)

Given a Java source code (as shown below) specifying the *seat selection* of a passenger in a Flight Management System based on the passenger type (e.g., *Economy, Flex, or Business*). Only *Flex* and *Business* passengers can select a seat. Please answer the following questions:

1. Which issue (in terms of the SOLID principles violation) could be triggered in the implementation? *(Name the principle and explain the issue)*
2. How to solve the issue? *(Provide a solution in the form of Java source code)*

Note: *Please create a PDF file and include all the answers to the questions.*

```java
abstract class Passenger {
    public abstract void selectSeat() throws Exception;
}

class Economy extends Passenger {
    @Override
    public void selectSeat() throws Exception {
        // Economy passengers cannot select a seat
        throw new Exception("Seat selection is not allowed!");
    }
}

class Flex extends Passenger {
    @Override
    public void selectSeat() {
        // statements regarding seat selection of Flex passengers
    }
}

class Business extends Passenger {
    @Override
    public void selectSeat() {
        // statements regarding seat selection of Business passengers
    }
}

class Main {
    public static void main() throws Exception {
        List<Passenger> passList = new LinkedList<>();
        passList.add(new Business());
        passList.add(new Economy());
        passList.add(new Flex());

        for (Passenger p : passList) {
            p.selectSeat();
        }
    }
}
```

Exercise 3 (Practical Part 1.3 = 3.5 Points)

Given a Java source code (as shown below) specifying a payment method that can be used in an online order session, please answer the following questions:

1. Which issue (in terms of the SOLID principles violation) could be triggered in the implementation? *(Name the principle and explain the issue)*
2. How to solve the issue? *(Provide a solution in the form of Java source code)*

Note: *Please create a PDF file and include all the answers to the questions.*

```java
interface IPayable {
    void pay();
}

class CreditCardPayment implements IPayable {
    public CreditCardPayment(double amount, String cardNumber, String cvv) {
        // statements for initializing variables
    }

    public void pay() {
        // statements to proceed a credit card payment method
    }
}

class BankTransferPayment implements IPayable {
    public BankTransferPayment(double amount, String iban, String pincode) {
        // statements for initializing variables
    }

    public void pay() {
        // statements to proceed a bank transfer payment method
    }
}

class Customer5 {
    IPayable payment;

    public void payOrder() {
        // statements to proceed the payment of an order
        // pay using a credit card
        payment = new CreditCardPayment(amount, cardNumber, cvv);
        payment.pay();
        // pay using a bank transfer
        payment = new BankTransferPayment(amount, iban, pincode);
        payment.pay();
    }
}
```

Exercise 3 (Practical Part 2 = 10 Points)

Assume an entrepreneur is running a restaurant which primarily serves different types of Japanese Ramens. She wants to have an Order Management System (OMS) that supports customers to order Ramens. For the first milestone, she has defined several requirements that need to be fulfilled by the OMS:

- The restaurant offers different types of Ramen. Every Ramen contains at least two ingredients *dashi* and *noodle*. It can be combined with additional toppings (e.g., Sea Food, Roasted Pork, Fish Cake, Spicy Ground Pork). Ramens with certain topping combinations are named, e.g., *Sapporo Ramen* for the base ramen with sea food.
- The system is able to show Ramens' information to customers, including *"name", "main ingredients",* and "*price per portion".* An example of Ramens' information is depicted in the table below.
- The system allows a customer to order a Ramen and the number of portions. Besides, in each order, a customer can choose different Ramens with corresponding number of portions. For instance, Sapporo Ramen x 02 (portions), Kitakata x 03 (portions), etc.
- After completing an order, the system should be able to print out a bill to the customer. The bill consists of all the order information such as chosen Ramens, number of portions for each, and total price.

Ramen	Main Ingredients	Price (Euro)/Portion
Sapporo Ramen	Dashi, Noodle, See food	15,99
Hakata Ramen	Dashi, Noodle, Roasted Pork	14,99
Kitakata Ramen	Dashi, Noodle, Fish cake	13,99
Nagoya Taiwan Ramen	Dashi, Noodle, Spicy Ground Pork	12,99

Your tasks:

- ✓ Use the design pattern(s) you have learnt in the lecture topic "*Design Patterns*" to design and implement the mentioned functionalities of the OMS.
- ✓ Create a corresponding UML class diagram that illustrates your design and include it in a PDF file.
- ✓ Implement the mentioned functionalities using Java (developing UIs is not mandatory). Provide the implementation in *.zip* source code file and executable *jar* file.

Submission of Exercise 3

- **Theoretical part**: NO submission needed (the points are calculated from the share of your correct answers in https://checkr.tugraz.at/.
- **Practical part**: Submit one integrated .zip file to TeachCenter that includes the solutions to the practical part.
- **Deadline (strict) for Exercise 3: 14:00, 25.06.2021**

Thank You!

In case of questions, please

contact the newsgroup: tu-graz.lv.oad