Design Patterns Exam

The exam has two parts:

1. First part, paper exam with a multiple choice questions and UML modeling (14 points)
2. Second part, coding exam (6 points) (questions 3 and 6). For each exercise, it is expected to submit the code and small document that describes how you have solved the problem. For instance, which design patterns you have used, which methods or which classes you have created, etc.

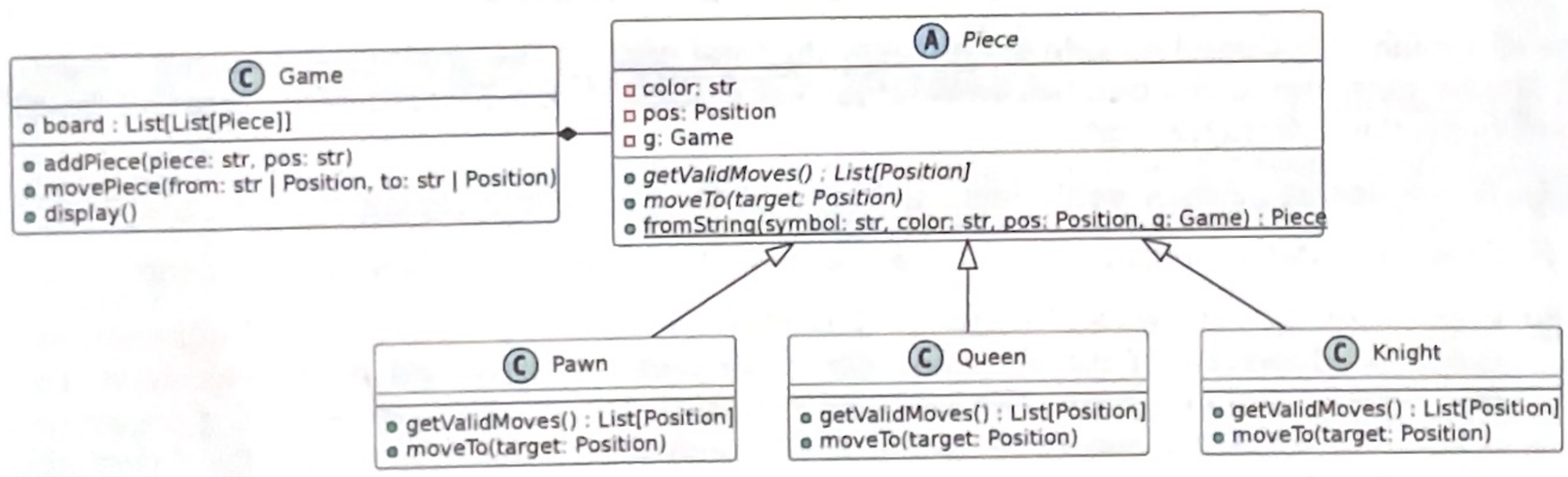
You can start the coding exam after finishing the first part (you have to give back the paper exam before starting the coding exam).

At the end of the coding exam, you should submit a zip file with the code and the documents that you have created, with one folder by exercise. The zip file should be named with your name and the date of the exam, for instance, `john_doe.zip` and its content should be like this:

```
john_doe.zip
├── exercise1
│   ├── ...
│   └── README.md
├── exercise2
│   ├── ...
│   └── README.md
```

Exercise 1: Pown promotion

Consider the following current UML representation of the classes of our chess game (simplified for the question).



`moveTo` is used to move a piece to a new position. The `getValidMoves` method returns a list of valid positions where the piece can move. The `addPiece` method is used to create and add piece to the board to a given position in algebraic notation (e.g., "a1"). The `movePiece` method is used to move a piece from one position to another in algebraic notation (e.g., "a1" to "a2"). The `display` method is used to display the board.

1) In chess, a pawn can be promoted to another kind of piece when it reaches the opposite side of the board. One of your colleagues suggests implementing this feature by adding

extra code to the `Game` class (in `play`) However, you think that this is not the best approach, as it is not its responsibility to manage specific piece behavior, and should be instead in the `Piece` class.

Describe the difficulties of implementing the promotion feature. Which design pattern would be the most suitable to implement the promotion feature? (1pt)

2) Change (draw) the UML to be able to implement the promotion feature with the chosen design pattern. (3pt)

3. Implement the `promote` feature in the `moveTo` method of the `Pawn` class. The program should ask the user which piece he wants to promote the pawn to. The valid options are `Q` (Queen), `R` (Rook), `B` (Bishop), and `N` (Knight). **Some code is already provided to help you.** (3pt)

Exercise 2: IKEA

You are tasked with developing a software system for an IKEA online store. One of the key components of this system is the shopping cart, which needs to handle a variety of products. Some products are simple items, such as the

Product	Price
METOD Base cabinet frame, white	$32.00
METOD door white 80x37 cm	$144.00
MITTLED LED kitchen worktop lighting strip	$17.00
STENSUND Door, 60x80 cm	$33.00

But, you can also buy *bundles* of products, which are a combination of several products that are sold together to prevent the user from buying the wrong items. For instance, the bundle "Kitchen base set" contains the following products:

- 3 × METOD Base cabinet frame, white
- 3 × METOD door white 80x37 cm
- 1 × MITTLED LED kitchen worktop lighting strip

The sets can be also combined with other sets, for instance, the "Kitchen base set" can be combined with the "Kitchen tap & sink set" to form the "Kitchen set".

The shopping cart should be able to calculate the total price of the products in the cart, to list all *simple* items (as if the bundles were *unpacked*), and the user should be able to add and remove products from the cart.

4) Which design pattern would you use to model the products involved ? (1pt)

5) Draw the UML diagram of the classes involved in the shopping cart system. (3pt)

6) Implement the classes and methods described in the UML diagram. Add unittests to check the behavior of the shopping cart. **No code is provided here, you have to implement and structure the code by yourself**. Here is an example of the expected usage. Note that the usage of typing, object modeling decorators (eg *abstractmethod*) will be considered in the evaluation. Do not forget to provide a small document that describes how you have solved the problem. (3pt)

Factory.
Abs' Factory.
Prototype
compos.te
strategy.
state.