Code of Jigsaw Puzzle

Write the object-oriented code to implement the design of the jigsaw problem.

We'll cover the following

- Jigsaw puzzle
  - Enumerations
  - Side
  - Piece
  - Puzzle
  - Puzzle solver
- Wrapping up

We've gone over the different aspects of the jigsaw puzzle and observed the attributes attached to the problem using various UML diagrams. Let's explore the more practical side of things, where we will work on implementing the jigsaw puzzle using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the jigsaw puzzle:

- Java
- C#
- Python
- C++
- JavaScript

Jigsaw puzzle

In this section, we'll provide the skeleton code of the classes designed in the class diagram lesson.

> **Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

Enumerations

The following code defines the `Edge` enum that represents the shape of the puzzle piece:

> **Note:** JavaScript does not support enumerations, so we will be using the `Object.freeze()` method as an alternative that freezes an object and prevents further modifications.

```
enum Edge {
  INDENTATION,
  EXTRUSION,
  FLAT
}
```

Enum definitions

Side

The `Side` class contains a reference to the `Edge` enum that will describe the edge shape of any side of a puzzle piece. Its definition is given below:

```
public class Side {
  private Edge edge;
}
```

The Side class

Piece

The `Piece` class contains a list of sides (representing each side of a piece) and ensures if a particular piece is a corner, middle, or edge using its respective functions. The definition of this class is provided below:

```
public class Piece {
  private List<Side> sides = Arrays.asList(new Side[4]);
  public boolean checkCorner() {}
  public boolean checkEdge() {}
  public boolean checkMiddle() {}
}
```

The Piece class

Puzzle

The Puzzle class includes the entire board of the jigsaw puzzle as well as the unused pieces. Since the board is a rectangle, it will be represented using a 2-D array. In addition, there will be only one instance of the puzzle board in the jigsaw puzzle. Therefore, the Puzzle class will be a Singleton class to ensure that only one instance for the board is created in the entire system. The definition of this class is given below:

```
public class Puzzle {
  private List<List<Piece>> board;
  private List<Piece> free; // represents the currently free pieces (yet to be inserted in board)
  public void insertPiece(Piece piece, int row, int column) {};

  // Puzzle is a singleton class that ensures it will have only one active instance at a time
  private static Puzzle puzzle = null;

  // Created a static method to access the singleton instance of Puzzle class
  public static Puzzle getInstance() {
      if (puzzle == null) {
          puzzle = new Puzzle();
      }
      return puzzle;
  }
}
```

The Puzzle class

Puzzle solver

The `PuzzleSolver` class tries to solve the puzzle using the `matchPieces()` function. Its definition is given below:

```
public class PuzzleSolver {
  public Puzzle matchPieces(Puzzle board) {}
}
```

The PuzzleSolver class

Wrapping up

We've explored the complete design of a jigsaw puzzle in this chapter. We've looked at how a basic jigsaw puzzle can be visualized using various UML diagrams and designed it using object-oriented principles and design patterns.

Back

Class Diagram for the Jigsaw Puzzle

Complete

Next

Getting Ready: The Airline Management System