Programming Practice

9 October 2024

Problem 1: Shape Hierarchy

Design and implement a hierarchy of shape classes that demonstrates inheritance and polymorphism. Your implementation should include:

1. An abstract class `Shape` with:
   a. An abstract method `double getArea()`
   b. An abstract method `String getDescription()`
2. Concrete classes extending `Shape`:
   a. `Circle` (field: radius)
   b. `Rectangle` (fields: width, height)
   c. `Triangle` (fields: base, height)
3. Implement the `Comparable <Shape>` interface in the `Shape` class to compare shapes based on their areas.
4. Create a utility class `ShapeAnalyzer` with a static method:

```
public static <T extends Shape> T findLargestShape(List<T> shapes)
```

   This method should return the shape with the largest area from the given list.
5. In the `main` method:
   a. Create a list of different shapes;
   b. Use the `ShapeAnalyzer` to find and print the largest shape;
   c. Sort the list of shapes and print them in ascending order of area.

Ensure proper encapsulation, use appropriate access modifiers, and include necessary constructors and getter methods.

Provide at least one test case for each of the above methods.

Problem 2: Generic Stack with Iterator

Implement a generic stack data structure with an inner iterator class. Your implementation should include:

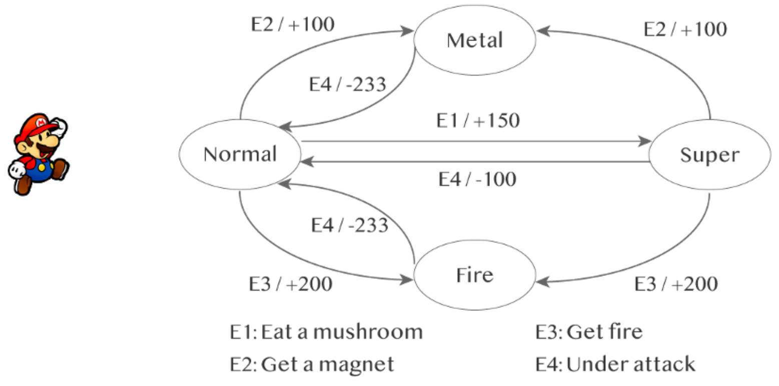
1. A generic class `Stack<T>` with the following methods:
   a. `void push(T element)`

   b. `T pop()`
   c. `T peek()`
   d. `boolean isEmpty()`
   e. `int size()`
2. An inner class `StackIterator` implementing the `Iterator<T>` interface with methods:
   a. `boolean hasNext()`
   b. `T next()`
   c. `void remove()` (optional: throw UnsupportedOperationException if not implemented)
3. In the `main` method, demonstrate the usage of your `Stack<T>` class:
   a. Create a `Stack<String>` and insert several elements;
   b. Use a for-loop to iterate through the stack and print its elements;
   c. Create a `Stack<Integer>` and insert several elements;
   d. Use the `StackIterator` explicitly to iterate through the stack and print its elements.

Ensure proper exception handling (e.g., for pop/peek on an empty stack) and include appropriate comments to explain your code. Provide at least one test case for each of the above methods.

Problem 3: State Machine in Super Mario Bros

Similar to the state machines involved in compilation principles, characters in games often change their states as triggered by various events. As illustrated in the figure below, in the famous game named *Super Mario Bros*, a character is initially in the *Normal state*. It can enter the *Super state* when it eats a mushroom, and will receive 100 points at the same time. An attack will make it return to the *Normal state* and lose a certain amount of points. Garfield, a cat who loves playing *Super Mario Bros*, asks you to implement the state transition mechanism as shown in the figure below.



Design and implement one or multiple public classes representing the various states of Mario, where the initial score is 0. Implement a series of public methods that correspond to the four events. In addition, provide methods to retrieve the current score and state of Mario.

Important notice: **Do NOT** use any if-else or switch statement in your code.

Provide at least one test case for each of the above methods.