Computer Science 2A

Practical Assignment Bonus 2

Assignment date:	2024-05-18
Deadline	2024-05-24 13h00
Marks: 100	

This practical assignment must be uploaded to eve.uj.ac.za **before** 2024-05-24 13h00. Late[1] or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of documentation. Marks will be deducted if these are not present. Every submission **must** include a batch file unless stated otherwise.

The **reminder page** includes details for submission. Please ensure that **ALL** submissions follow the guidelines. The reminder page can be found on the last page of this practical.

This practical aims to familiarise you with the ***Visitor Design Pattern***.

The **Utopian Artificial Intelligence Forum**[2] is struggling to add extra functionality to existing classes with minimal changes. Apparently, there is a way to visit these models to invoke functionality. They would like you to demonstrate this concept to calculate the area of Shapes.

You have been provided with a starter project that contains the following:

- a `PB2.jar` that contains:
  - Visitor Interfaces: `AbstractShapeVisitor`, `AbstractShapeVisitable`
  - a `PB2Tester` class: Used in the `Main` class to determine if your code will work.
- a `Main` class: DO NOT MAKE CHANGES TO THIS FILE!

You are required to make the following classes in the `acsse.csc2a.model` package:

- `Circle`: This class should have a constructor that takes a `double radius` and a getter method `getRadius()`.
- `Rectangle`: This class should have a constructor that takes `double width, double height` and getter methods `getWidth()` and `getHeight()`.
- `Triangle`: This class should have a constructor that takes `double base, double height` and getter methods `getBase()` and `getHeight()`.

[1] Alternate arrangements for exceptional circumstances will been posted on eve.

[2] Disclaimer - This series of problem statements are a work of fiction. Names, characters, businesses, places, events and incidents are either the products of the author's imagination or used in a fictitious manner. Any resemblance to actual persons, living or dead, or actual events is purely coincidental.

Additionally, you need to implement the following in the `acsse.csc2a.visitor` package:

- `ShapeAreaVisitor`: This class should implement the `AbstractShapeVisitor` interface and calculate the area for each shape. The methods are:
  - `visit(Circle circle)`: Calculates and prints the area of a circle.
  - `visit(Rectangle rectangle)`: Calculates and prints the area of a rectangle.
  - `visit(Triangle triangle)`: Calculates and prints the area of a triangle.

The output should look like the following:

```
Circle Area: 78.53981633974483
Rectangle Area: 24.0
Triangle Area: 6.0
```

Instructions

1. **Create Concrete Shape Classes:**
   - Implement `Circle`, `Rectangle`, and `Triangle` classes that implement the `AbstractShapeVisitable` interface.
   - Ensure the constructors for these classes match the specified signatures.
   - Implement getter methods for each class to retrieve their properties.
2. **Implement Concrete Visitor:**
   - Implement the `ShapeAreaVisitor` class to calculate the area of each shape and print the result.
3. **Test the Visitor Pattern:**
   - Use the provided `Main`[3] class to test your implementation. Do not make any changes to this file.

Mark sheet

1. Correct Implementation of the Visitor Design Pattern (Note this is an all or nothing practical. If you do not provide the correct output, you will not get the marks.) [100]

NB

Submissions which **do not compile** will be capped at 40%!

Practical marks are awarded subject to the student's ability to explain the concepts and decisions made in preparing the practical assignment solution. (Inability to explain code = inability to be given marks.)

Execution marks are awarded for a correctly functioning application and not for having related code.

[3]We may replace your Main, so your classes should all be implemented (not just the ones used in the Main class).

Reminder

Your submission must follow the naming convention below.

```
SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER
```

Example

Surname	Berners-Lee	**Module Code**	CSC02A2
Initials	TJ	**Current Year**	2024
Student number	209912345	**Practical number**	PBonus 2

```
Berners-Lee_TJ_209912345_ CSC02A2_2024_PBonus 2
```

Your submission must include the following folders:

Folder	**State**	**Purpose**
`bin`	*Required*	Should be empty at submission but will contain runnable binaries when your submission is compiled.
`docs`	*Required*	Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. All files must be in **PDF** format. Your details must be included at the top of any **PDF** files submitted. **Do not include generated JavaDoc.**
`src`	*Required*	Contains all relevant source code. Source code must be places in relevant sub-packages! Your details must be included at the top of the source code.
`data`	*Optional*	Contains all data files needed to run your solution.
`lib`	*Optional*	Contains all libraries needed to compile and run your solution.

NB

Every submission **must** include a batch file that contains commands which will:

- Compile your Java application source code.
- Compile the associated application JavaDoc.
- Run the application.

Do not include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.

Multiple uploads

Note that only **one** submission is marked. If you already have submitted once and want to upload a newer version then submit a newer file with the same name as the uploaded file in order to overwrite it.