S.O.L.I.D. Principles

My Learnings with this Task:

With this task I learnt extensively about the S.O.L.I.D. principles, and how to apply them in an application-based scenario. This helped me a lot in learning the practicality of these principles, and by doing hands on it has given me a clear understanding how to implement this in my projects.

Document Structure:

First each of the violations and corrections for the S.O.L.I.D. principles are explained, followed by the references section. Lastly, *At the end of this document is the code output attached.*

This report contains an explanation of the Bad and Good packages for each of the principles.

1. Single Responsibility Principle:

Description/ Problem context scenario:

I have chosen a Quiz application scenario, having Questions and Answers, with two options. Moreover, the grading aspect of an attempted quiz like: Percentage marks obtained, and letter grades obtained is also implemented. Additionally, these results need also to be saved for persistence and easy lookup by the professor, hence the quiz result saving to file functionality is also included. The inspiration for this scenario has come from my "Quizly-Test-Driven-Development-Java-Junit" where I created a Quiz application for professors and students. [1]

Bad:

Classes:

- **QuestionQuiz**: This class is created to hold the Question, Answer, and Options details. The object of this class represents a single logical Question-Answer present within a Quiz.

- **Quiz**: This class is the main manager class, which manages all quiz operations, which include the following functionalities:
    - **loadQuestions**: This is responsible for loading the multiple Quiz Questions and Answers.
    - **displayQuestion**: This function takes in user input corresponding to each Question, and option and calculates the score. This helps users to attempt the quiz.
    - **getPercentage**: This function calculates the percentage score obtained by the user after successfully completing the quiz.
    - **getLettergrade**: This is responsible for calculating the letter grade of the quiz attempted successfully by the user based on the percentage obtained.
    - **displayResult**: This is responsible for displaying the Quiz's final result after the user answers all questions completely.
    - **saveResult**: This function has the responsibility to save the user's quiz result to a file, with all the attributes defining that quiz.

Why is this BAD/Violating the Single Responsibility Principle?

The reason for this is that the class named: 'Quiz', has methods defined which is responsible for multiple tasks. The Single Responsibility Principle (**SRP**) states each class should fulfil only one responsibility. In this case, the Quiz class is responsible for:

1. Loading or retrieval of Quiz question answers from the data repository.
2. Displaying (console prompts and display) and functionality to Attempt Quiz
3. Calculating Grades and Evaluating Quiz Results Performance
4. Saving Results to file in the directory.

Hence, clearly, this is a violation of SRP. To fulfill the SRP the single Quiz class needs to be split or divided across multiple classes, such that each class fulfills a single responsibility. Let's have a look below at the Corrected scenario.

Good:

Classes:

1. **QuestionQuiz**: This class is created to hold the Question, Answer, and Options details. The object of this class represents a single logical Question-Answer present within a Quiz.

2. **QuizLoadData**: This class is responsible for loading the Quiz Questions and Answers from a data repository. (In my scenario I have instantiated the questionQuiz objects and created the Quiz)

3. **QuizManager**: This class is responsible for prompting the user to attempt a quiz by displaying Questions and options to the user. Additionally, it is also responsible for displaying performance results to users by communicating with the QuizEvaluator instance.

4. **QuizEvaluator**: This class is responsible for handling the Grading, and performance evaluation aspects of the successfully attempted quiz.

5. **QuizSaveResultsToDisk**: This class is responsible for saving the user's grades and quiz performance to file.

How did I fix the SRP violation?

The SRP violation is **corrected by splitting the main Quiz class** in the 'bad' package, **into 5 different classes, each having a single responsibility**.

So, as discussed in the above Classes section inside 'Good' each class is responsible for one task, And the Quiz Manager is the quiz driver class which uses the rest of the classes to emulate a Quiz application. To give an example about this, The Quiz Manager class uses QuizEvaluator private object: which is responsible for grading and evaluating the user's quiz performance based on the user's answers to quiz questions. Additionally, the *displayQuizQuestion* inside QuizManager class requires the list of all quizQuestions to be passed as an argument, such that the function prompts the user for the Question answer and tracks the users' quiz performance accordingly. Lastly, the displayResult

function in QuizManager is responsible for displaying the user's quiz Results with the help of the QuizEvaluator object.

Overall, there is no class which is having shared or multiple responsibilities coupled inside that class. Hence, each class is fulfilling only one responsibility.

Reference Note: In the code example the Data for Quiz question answers are taken from [2].

---

2. Open Closed Principle:

Description/ Problem context scenario:

I have chosen a restaurant billing order scenario where a customer can order menu items from a restaurant, and the restaurant order would track the menu items ordered by the customer. Lastly, it would also generate the billing invoice of the customer's order with the 15% HST tax applied to it.

The reason why I have chosen this specific scenario is that normally one might think the order is a single entity and doesn't require a further extension, which is wrong in a real-world context. This is because for example restaurants may also offer special discounts on weekends, cashback on certain order purchases, or restaurant coupons with restaurant-affiliated stalls. Hence, this proves that order is not just a simple order with standard bills, instead in the real world there are many extensions to it. Now looking at the Open Closed principle violation

Bad:

Classes:

1. **RestaurantMenuItem**: This class stores a Menu Item detail, like the item's name, price and its type (which is item cuisine or item type).

2. **RestaurantOrder**: This class represents the order of the customer, which holds the different menu items ordered by the customer. It has functions for adding menu Items in a List of menu items, generating the total order bill amount with tax applied to it.

Why is this BAD/Violating Open Closed Principle?

The reason why this violates the open-closed principle is because if I want to introduce a new feature, including weekly or festival day discounts or occasional cashback on certain menu item orders or limited-time coupons on customer dining. Then in this scenario based on the above class, it is not possible to incorporate these features, and it is not possible to add order-related features as well without modifying the existing class code base (which is the class named: RestaurantOrder), which violates the open-closed principle (OCP) as it is open for modification and greatly increases the risk of introducing bugs into the code base. As OCP says the software entities should be open for extension but closed for modification. [3] This can be corrected by creating a standard Restaurant Order Interface contract, and then extending the new features like special offer orders, and coupon orders by implementing this Restaurant order interface.

Now, let's have a look at the corrected code base which complies with open closed principle.

Good:

Interfaces:

1. **IRestaurantOrder**: Interface which holds the information about a standard order. In my case, it has two functions. First is the generation of the order's total bill amount, second is to display the invoice. Moreover, this class can be implemented by concrete classes to further extend functionalities.

Classes:

1. **RestaurantMenuItem**: This class stores a Menu Item detail, like the item's name, price and its type (which is basically item cuisine or item type).
2. **RestaurantOrder**: This class represents the standard basic restaurant order bill with tax calculation functionality. This class implements the interface IRestaurantOrder.
3. **SpecialOfferRestaurantOrder**: This class represents the extension of the standard basic Order bill, with cashback discount functionality included in it. This class also implements the interface IRestaurantOrder. The cashback logic is included inside the overridden *generateTotalOrderBillAmount* function from the interface.

How did I fix the OCP violation?

The violation can be corrected by creating an interface defining the common behaviour of basic standard restaurant Order related functionalities (in my case named: IRestaurantOrder). Then the different restaurant orders can implement this interface. I then created a class named: RestaurantOrder, which defines the basic Order. Then as a part of the extension, I created a SpecialOfferRestaurantOrder class which extends the IRestaurantOrder interface and incorporates the basic RestaurrantOrder class object inside it. This helps me to introduce a new special offer with a cashback feature, without making any modifications inside the concrete Restaurant Order class. Hence, this complies with the OCP as it allows extension without making any modifications to the existing code base.

NOTE: Additional higher-order thinking learning I learned from this example:

When looking at this specific scenario closely you will observe that there is a possibility, I can create this special offer restaurant order by extending the basic RestaurantOrder class. With this one may say I have the advantage of reusing the standard Restaurant order functionality, which is a plausible point to note. But, when thinking from a real-world profitability perspective, I would ideally want to make these special orders not directly depend on the basic restaurant order because: Firstly, if I am extending the basic order class then I must override the method implementation of the parent class, but I want to take advantage of the functionality in the basic order class as well. So, instead, I implement the interface, then use the object of the basic order class and use its function in addition to the special offer's order functionality. This helps me use both classes' functionality without having the risk of introducing unintended bugs in my special offer order when any modification is done in the basic restaurant order.

Reference Note: In the code example the Data for Menu items are taken from [2].

---

3. Liskov Substitution Principle:

Description/ Problem context scenario:

I have chosen an event organization scenario, where I have different types of events like Musical Concert events, technical or cultural Exhibitions, and Business conferences. In my scenario, I am trying to generate the schedule of an event based on the event type, as each event type has a different event schedule structure. So, the aim in this problem scenario is should be able to generate the schedule of an event without requiring changes to existing code. Allowing easy extensibility without affecting existing functionality. Hence, I show a code example which violates the Liskov Substitution Principle (**LSP**), and the corrected example. By complying LSP it will ensure that the software entities can handle any specific event implementations interchangeably without any conditional logic.

Bad:

Classes:

1. **EventSesion**: This is a class which represents a single simple logical session of an event. An example for this to understand is: I can have a session called "Introduction to IBM HR managers" as the first session of a business conference event called "IBM showcase".
2. **Event**: This class represents the event entity, having multiple event sessions inside it for different event Types. It has the functionality of generating an event schedule based on the event type.

Why is this BAD/Violating the Liskov Substitution Principle?

The reason why this code example scenario violates LSP is because the method *generateSchedule* in the Event class checks the type of event and then performs the action based on the event type. This violates the LSP the specific behaviour is assumed based on the event type. Moreover, adding a new event type for example: Technical and Meditation 'Workshops' would require the modification of the *generateSchedule* method in the Event class, which violates the Liskov Substitution principle.

Good:

This example is an extensive problem as it violates LSP. Here I have two versions which satisfy LSP. The first solution:

Package Name: **LSP**

Abstract class:

- **Event**: This abstract class represents the abstract event entity, having the basic structure common to any event. It has an abstract method signature for generating an event schedule.

Classes:

1. **EventSesion**: This is a class which represents a single simple logical session common for any event.
2. **Exhibition**: This concrete class represents the cultural or technical Exhibition event entity type. This class extends the abstract class Event. This class has a custom implementation for the *generateSchedule* method specific to the Exhibition entity.
3. **Conference**: This concrete class represents the business Conference event entity type. This class extends the abstract class Event. This class has a custom implementation for the *generateSchedule* method specific to a Conference entity.
4. **Concert**: This concrete class represents the Musical or DJ Concert event entity type. This class extends the abstract class Event. This class has a custom implementation for the *generateSchedule* method specific to the Concert entity.

How did I fix the LSP violation?

This violation is corrected by creating an abstract class 'Event' having the common structure and interface for all event types. This abstract class has an abstract generateSchedule method definition. This abstract function has to be implemented by any event that extends this abstract class, in our case following classes extend this abstract class: Conference, Concert, and Exhibition classes. Each of these subclasses overrides this abstract method and implements its event-specific behaviour for schedule generation. Hence, the final design where the single concrete Event class is separated into event-specific classes-Conference, Exhibition, and Concert class, it complies with the LSP, by ensuring that all the subclasses can be used interchangeably with the base class without changing the correctness of the program. Now, as can be seen in the Main function, I can call the *generateSchedule* function of the abstract class object by instantiating it with the event-specific classes and calling the common method, which promotes extensibility and flexibility in this Event organization scenario.

Additionally:

When looking at the package name: **LSP_v2**

Here when looking at EventSession class although this is common across the different event types, it doesn't follow a good design principle. Moreover, it can be said that although very little somewhat raises concerns about it violating LSP. So the solution for this is simple. As seen above with the Event entity, here for the Session entity I created an EventSession abstract class. This abstract class is then implemented by the Event type-specific session classes like: ConferenceSession, ExhibitionSession and ConcertSession. By doing this it promotes the advantages that LSP promotes by adhering to this principle.

Reference Note: In the code example the Data for Concert is taken from [5], the data for the Exhibition is taken from [6], and the data for the Conference is taken from [7].

4. Interface Segregation Principle (ISP):

Description/ Problem context scenario:

For this scenario, I have chosen to apply the concept of the Interface segregation principle to SpaceX, a spacecraft engineering company. As the company deals with space crafts for exploring the planet Mars etc, I have chosen two space craft types: Rover, and Starship rocket spaceship. The responsibility of the rover is to launch, fly, land, communicate with Earth, and find/locate places where resources or minerals are available for analysis. And starship rocket has the responsibility to launch, fly, land, communicate with Earth, collect samples from a location, and analyze the collected samples. Let's see how the violated and corrected ISP code examples are.

Bad:

Interface:

1. **Spacecraft:** This interface is responsible for creating the blueprint of a standard spacecraft. This includes method signatures for launch, fly, land, collectSamples, analyzeSamples, communicate, locateSampleSource.

Classes:

1. **SpaceXRover**: This concrete class represents the Rover spacecraft of spaceX. This class implements the interface Spacecraft and overrides the method for implementing the Rover-specific functionalities.
2. **SpaceXStarship**: This concrete class represents the Starship spacecraft of spaceX. This class implements the interface Spacecraft and overrides the method for implementing the Starship rocket spacecraft-specific functionalities.

Why is this BAD/Violating Interface Segregation Principle?

The reason why this example is violating ISP is that there are few methods in the Spacecraft interface which are not needed to be implemented by all the spacecraft. In this specific example, Rover is not responsible for collecting and analyzing samples. Hence when the Rover class implements this interface it has to compulsorily override these methods which are not a part of the Rover concrete class. Similarly with Starship whose responsibility is to collect and analyze samples but NOT locate sample sources, thus again a method is being implemented which it does not use or represent that concrete class. Hence, violating the ISP.

The solution to this is to segregate the single common interface Spacecraft into multiple specific interfaces. Let’s have a look at it below.

Good:

Interfaces:

1. **BasicSpacecraft:** This interface includes the basic functionality or methods defining a basic spacecraft. This includes functions to: launch, fly, and land a spacecraft.
2. **ExplorerSpacecraft:** This interface includes the functionality performed by a spacecraft that is responsible for exploration. Currently, this includes methods for locating sample mineral sources.
3. **AnalyzingEngineSpacecraft:** This interface includes additional advanced functionality of a spacecraft to perform analysis. The methods in this interface include: collecting samples and a method to analyzeSamples.
4. **CommunicationSpacecraft:** This interface is responsible for the different communication functionalities that a spacecraft may include. Currently, it has a method called communicate, which is for communicating with Earth. *For a more extensive scenario,* it can also include a communication method for communicating between Rovers or Starships or even for communicating coordination messages among spacecraft.

Classes:

1. **SpaceXRover**: This concrete class represents the Rover spacecraft of spaceX. This class implements the interfaces BasicSpacecraft, CommunicationSpacecraft, and ExplorerSpacecraft. It implements a BasicSpacecraft interface so the rover can have the basic functionalities for movement in space. It implements CommunicationSpacecraft because it needs to send messages to Earth about resource locations. Lastly, it implements ExplorerSpacecraft because it needs to find or explore new resource locations. *This exploration is done only by Rover and not by Starship.*

2. **SpaceXStarship**: This concrete class represents the Starship spacecraft of spaceX. This class implements the interfaces BasicSpacecraft, AnalyzingEngineSpacecraft, and CommunicationSpacecraft. The reason for implementing BasicSpacecraft and CommunicationSpacecraft is so that Starship has basic movement and communication capabilities respectively. Lastly, the reason for implementing AnalyzingEngineSpacecraft is that Starship can collect samples from a location, and analyze them. *This analysis is done only by Starship and not Rover.*

How did I fix the ISP violation?

ISP violation is adhered to by separating the SpaceCraft interface into multiple smaller interfaces with a single responsibility defining them. In this scenario the interface is separated based on the following responsibility or spacecraft feature:

- Movement: **BasicSpaceCraft**
- Communication: **CommunicationSpacecraft**
- Exploration of Resource: **ExplorationSpacecraft**
- Analysis of Resource samples: **AnalyzingEngineSpacecraft**

And, then in the 'bad' we saw how the Rover and Starship concrete classes were overriding methods which did not represent their capability. Now, by segregating the interface and implementing only those interfaces that define the concrete class capabilities, we do not have any method which is being implemented unnecessarily and irrelevant to that specific class.

So here since Rover implements BasicSpacecraft, CommunicationSpacecraft, and ExplorerSpacecraft. The unnecessary methods named: "collectSamples", and "analyzeSamples" are not implemented now in the 'Good' package.

Similarly, Starship since it implements BasicSpacecraft, AnalyzingEngineSpacecraft, and CommunicationSpacecraft. The unnecessary method named: "locateSampleSource" is not implemented in the "Good" package.

Hence, by segregating the interfaces based on the specific functionality the ISP is followed, allowing us enhanced reusability, maintainability, and flexibility. *The other best part about this is that it supports the use of dependency inversion by depending on abstractions.*

Reference Note: The inspiration for this scenario I was inspired from [8].

---

5. Dependency Inversion Principle (DIP):

Description/ Problem context scenario:

I have chosen a payment gateway situation where an e-commerce website has to manage, and track payment transfers or transactions on the platform. These payment transfers can be done via multiple agents like bank-specific agents like SBI, CIBC, ICICI, etc or popular platforms like Google Pay, Apple Pay or PayPal. Since we can have multiple payment agents that an e-commerce website can support. It is imperative that the code that is designed should be able to handle this, and should be extensible such that the user can use any of the payment agents without any issue with minimum coupling.

Bad:

Classes:

1. **PaymentTransaction**: This class includes the basic Payment Transfer Transaction details like the date and time of when the transaction occurred, transaction ID, the amount transferred, transaction remarks, and payment status.
2. **SBIPaymentAgent**: This class represents the State Bank of India (SBI) bank-specific agent implementation, which utilizes PaymentTransaction for storing, tracking and managing transactions. The methods include initiating Interac money transfers, processing payment transfers, refunding payment transfers, displaying transaction history, total money transfers processed in a day, and the number of payments processed in a day.
3. **PaymentGateway**: This can be considered as the main Payment Gateway controller which controls the different payment agents for payment transfer on the e-commerce platform. Currently, in my example, I have taken a single payment agent for payment tracking by agent.

Why is this BAD/Violating Dependency Inversion Principle?

The reason this example violates DIP is because the concrete class “PaymentGateway” is highly dependent on the “SBIPaymentAgent” concrete class, which creates tight coupling among these classes. Moreover, when looking at the implementation for handling the agent-specific payment transfer functions, All of these functions are invoked on the specific PaymentAgent class object.

Hence, This violates DIP, as DIP states that the high-level and low-level modules should depend on abstractions and not on actual concrete modules.

To solve this issue an interface for Payment Agents can be created which has the basic functionalities which are common across all the payment agents. Then each of the concrete agent classes can implement this interface and override the methods with their agent-specific payment transfer logic. Let's have a look below:

Good:

Interfaces:

1. **PaymentManager**: This is the interface which has the common functionalities that each Payment Agent should possess to perform the action on transactions.

Classes:

1. **PaymentTransaction**: This class represents the single logical payment transaction unit. This includes basic Payment Transfer Transaction details like the date time of when the transaction occurred, transaction ID, the amount transferred, transaction remarks, and payment status.

2. **SBIPaymentAgent**: This class represents the State Bank of India (SBI) bank-specific agent implementation, which utilizes PaymentTransaction for storing, tracking and managing transactions. This class implements the PaymentManager interface for implementing the agent-specific logic for transaction processing. The methods include initiating Interac money transfers, processing payment transfers, refunding payment transfers, displaying transaction history, total money transfers processed in a day, and the number of payments processed in a day.

3. **PaymentGateway**: This can be considered as the main Payment Gateway controller which controls the different payment agents for payment transfer on the e-commerce platform. Here, the Payment Agent is injected via its constructor.

How did I fix the DIP violation?

The violation is corrected by creating an interface PaymentManager which has the basic methods that any Agent should possess for performing the action on Transaction. I then create one Agent named: SBIPaymentAgent, this agent implements the above interface for agent-specific transaction operations. Then in the main PaymentGateway class, the SBIPaymentAgent class is injected through its constructor, allowing dependency injection. This allows the PaymentGateway class to use different Payment Agent class implementations without being tightly coupled to a specific agent’s class object. Moreover, now there is no need to make changes to the PaymentGateway class whenever a new agent is added to the system. Hence, this is how the violation of the DIP is corrected.

The best part about this is that it promotes flexibility and easy switching between different payment agents. Moreover, it promotes loose coupling, and code modularity because now the PaymentGateway class interacts with the PaymentManager interface, which can further be implemented by other concrete Agent classes like Google Pay or Apple Pay e.tc

Note: For this task regarding Dates operations, and Random functions in Java, I learnt from [9],[10], and [11].

References:

[1] "Quizly-Test-Driven-Development-Java-Junit," G*itHub*, https://github.com/Zain-Saiyed/Quizly-Test-Driven-Development-Java-Junit/tree/main (accessed Dec. 16, 2023)

[2] "Class 11 Maths Chapter 9 Sequences and Series MCQs (With Answers)," *BYJUS*. https://byjus.com/maths/class-11-maths-chapter-9-sequences-and-series-mcqs/ (accessed Jul. 07, 2023).

[3] "SOLID: The First 5 Principles of Object Oriented Design", DigitalOcean. https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (accessed Dec. 16, 2023)

[4] "Mezza Restaurant," *canada2.givex.* https://wwws-canada2.givex.com/cws5/mezzapwa/#/online-ordering/menu/subcategory/6684 (accessed Jul. 07, 2023).

[5] "Dartmouth Sunshine Series," *halifax*. https://www.halifax.ca/parks-recreation/events/dartmouth-summer-sunshine-free-concerts (accessed Jul. 07, 2023).

[6] "Annapolis Valley Exhibition," *annapolisvalleyexhibition*. https://annapolisvalleyexhibition.com/ (accessed Jul. 07, 2023).

[7] "Conference Agenda Powerpoint Ppt Template Bundles," *slideteam*. https://www.slideteam.net/conference-agenda-powerpoint-ppt-template-bundles.html (accessed Jul. 07, 2023).

[8] SpaceX, "SpaceX," *SpaceX*. https://www.spacex.com/(accessed Jul. 07, 2023).

[9] "LocalDateTime (Java Platform SE 8)," docs.oracle. https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html. (accessed Jul. 07, 2023).

[10] "Random (Java Platform SE 8)," Oracle.com. https://docs.oracle.com/javase/8/docs/api/java/util/Random.html. (accessed Jul. 07, 2023).

[11] "Java LocalDate - Javatpoint," javatpoint. https://www.javatpoint.com/java-localdate. (Accessed Jul. 07, 2023).

Code Output:

BAD:

1. SRP:

```
================================================================================================================
SRP : Quiz application scenario where quiz class does answer evaluation, input tasks, and saving quiz result to file
================================================================================================================
*** Failing Single Responsibility Principle
Questions loaded successfully!

Question : If a, 4, b are in Arithmetic Progression; a, 2, b are in Geometric Progression; then a, 1, b are in:
Option 1 : H.P
Option 2 : A.P

Choose your option [1/2]:
1

Question : If a, b, c are in arithmetic progression, then:
Option 1 : 2b = a+c
Option 2 : b = a+c

Choose your option [1/2]:
1

Question : The sum of arithmetic progression 2, 5, 8, …, up to 50 terms is:
Option 1 : 3775
Option 2 : 3757

Choose your option [1/2]:
1

Question : Which of the following is an example of a geometric sequence?
Option 1 : 9, 20, 21, 28
Option 2 : 1, 2, 4, 8

Choose your option [1/2]:
1
```

```
Question : The next term of the given sequence 1, 5, 14, 30, 55, … is:
Option 1 : 91
Option 2 : 96

Choose your option [1/2]:
1
==== Quiz Completed Successfully! ====

==== Congratulations! ====
Quiz completed Successfully! Here are the results:
Total Questions    : 5
Correct Answers    : 3
In Correct Answers : 2
Percentage obtained: 60
Letter Grade       : C

==== Saving Quiz Result to file ====
[**SUCCESS**] Quiz Result saved successfully!
```

2. OCP:

```
============================================================================================================
OCP : Restaurant Billing Order scenario where for Festivals or Special offer class doesn't offer Extensibility
============================================================================================================
*** Failing Open Close Principle

============================================================
Scenario: Standard Billing Order after customer order
============================================================
---------- INVOICE ----------
Order date    : Sun Jul 09 09:40:40 UYT 2023
Order ID      : BILL_2835

----------------------
Ordered Items :
----------------------
Chicken Shawarma Plate  (Middle-Eastern cuisine)   ~~~~~~~~    $16.79
Mixed Grill Plate  (Middle-Eastern cuisine)   ~~~~~~~~    $18.89
8oz Hummus  (Middle-Eastern cuisine)   ~~~~~~~~    $7.99
Baklava  (Middle-Eastern dessert)   ~~~~~~~~    $7.96
Coca-Cola  (Cold Drink)   ~~~~~~~~    $5.98

Payment Method: Credit Card
----------------------------
SubTotal      : $57.61
HST 15.000%   : $8.0
Total         : $66.2515
----------------------------
```

3. LSP:

```
============================================================
LSP : Event Schedule generation for different type of events
============================================================
*** Failing Liskov Substitution Principle

============================================================
Scenario 1:' Concert ' Event type
============================================================
Concert Schedule:
------------------------------
Name    : Dartmouth Summer Sunshine Free Concerts
Date    : 2023-07-08
------------------------------
Time    : 2:00 PM - 3:00 PM
Artist  : Maura Whitman
------------------------------
Time    : 3:00 PM - 4:00 PM
Artist  : Brooklyn Blackmore
------------------------------
```

```
============================================================
Scenario 2:' Exhibition ' Event type
============================================================
Exhibition Schedule:
-----------------------------
Name    : Annapolis Valley Exhibition
-----------------------------
Session : Scarecrow contest
Time    : 10:00 AM - 3:00 PM
Date    : 14 August, 2023
-----------------------------
Session : Scarecrow contest
Time    : 10:00 PM - 3:00 PM
Date    : 15 August, 2023
-----------------------------
Session : Riverside stage
Time    : 8:30 PM - 9:45 PM
Date    : 19 August, 2023
-----------------------------
```

```
============================================================
Scenario 3: ' Conference ' Event type
============================================================
Conference Schedule:
-----------------------------
Name    : Human Resource Management Conference
Date    : 2023-08-15
-----------------------------
Session : Registration and Welcome
Time    : 7:30 AM - 12:00 PM
Speaker : Chair person
-----------------------------
Session : Pre-conference session
Time    : 8:15 AM - 11:15 AM
Speaker : Chair person
-----------------------------
Session : Lunch Break
Time    : 1:30 PM - 2:00 PM
Speaker : Organising team
-----------------------------
Session : Socialising strategy
Time    : 2:00 PM - 3:15 PM
Speaker : Zen Sai
-----------------------------
Session : CLosing ceremony
Time    : 3:15 PM - 3:30 PM
Speaker : Chair person
-----------------------------

Process finished with exit code 0
```

4. ISP:

```
================================================================================================
ISP : SpaceX spacecraft resource exploration scenario
================================================================================================
*** Failing Interface Segregation Principle

================================================================================================
SpaceX : Rover - Responsible for ONLY locating resource and NOT analysing resource
================================================================================================

--- FLY ---
Rover has already landed at base: Home Base and is Charging.
Possible options: Finalize the new location to search for possible resources, and launch Rover.
--- FLY ---


--- FINDING SAMPLE SOURCE ---
Rover is at base: Home Base and is Charging.
Rover Ready for launch at a new destination!
--- FINDING SAMPLE SOURCE ---


--- LAUNCH ---
Initiating launch, to search samples in area: Destination A312
Launching Rover towards area: Destination A312
--- LAUNCH ---


--- FLY ---
Rover is finding possible resource minerals at location: Destination A312
--- FLY ---
```

```
--- FINDING SAMPLE SOURCE ---
Rover is in flight, locating resource minerals...
...Found Collected sample num-[1]
Analysis Result: rich
...Found Collected sample num-[2]
Analysis Result: median
...Found Collected sample num-[3]
Analysis Result: sparse
...Found Collected sample num-[4]
Analysis Result: sparse
...Found Collected sample num-[5]
Analysis Result: rich
Total Mineral Resources found: 5
--- FINDING SAMPLE SOURCE ---

INVALID OPERATION! Do you mean to explore and find sample resource location? Consider: locateSampleSource()
INVALID OPERATION! Do you mean to explore and find sample resource location? Consider: locateSampleSource()

--- LAND ---
Rover docked at Base: Charging docking station-X22. Initiating charging.
--- LAND ---


--- FLY ---
Rover is finding possible resource minerals at location: Destination A312
--- FLY ---
```

```
--- COMMUNICATION ---
...Establishing connection with Earth...
...Sending sample's analysis result report to Earth...

--- FINDING SAMPLE SOURCE ---
Rover is in flight, locating resource minerals...
...Found Collected sample num-[1]
Analysis Result: sparse
...Found Collected sample num-[2]
Analysis Result: median
...Found Collected sample num-[3]
Analysis Result: rich
...Found Collected sample num-[4]
Analysis Result: sparse
...Found Collected sample num-[5]
Analysis Result: rich
Total Mineral Resources found: 10
--- FINDING SAMPLE SOURCE ---


==========================================================================================
SpaceX : Starship - Responsible for ONLY analysing resource and NOT locating resource
==========================================================================================

--- FLY ---
Starship has already landed at base: Home Base
Possible options: Check, Finalize operations, and initiate launch to another base destination.
--- FLY ---

INVALID OPERATION! Do you mean to analyze resource location? Consider: collectSamples() OR analyzeSamples()

--- LAUNCH ---
Initiating launch towards destination: Destination R998
Launching Starship towards destination: Destination R998
--- LAUNCH ---


--- FLY ---
Starship in flight towards base destination: Destination R998
--- FLY ---
```

```
INVALID OPERATION! Do you mean to analyze resource location? Consider: collectSamples() OR analyzeSamples()

--- SAMPLE COLLECTION ---
15
Starship in flight. Initiating sample collection...
...Collected sample [1]
...Collected sample [2]
...Collected sample [3]
...Collected sample [4]
...Collected sample [5]
...Collected sample [6]
...Collected sample [7]
...Collected sample [8]
...Collected sample [9]
...Collected sample [10]
...Collected sample [11]
...Collected sample [12]
...Collected sample [13]
...Collected sample [14]
...Collected sample [15]
Total samples collected: 15
--- SAMPLE COLLECTION ---


--- SAMPLE ANALYSIS ---
Starship has already landed at base: Home Base . Samples can be analysed while in flight for better analysis profitability.
--- SAMPLE ANALYSIS ---
```

```
--- LAND ---
Initiating Starship landing at Base: Charging docking station-X1A
--- LAND ---


--- FLY ---
Starship in flight towards base destination: Destination R998
--- FLY ---


--- COMMUNICATION ---
...Establishing connection with Earth...
...Sending sample's analysis result report to Earth...

--- SAMPLE COLLECTION ---
20
Starship in flight. Initiating sample collection...
...Collected sample [1]
...Collected sample [2]
...Collected sample [3]
...Collected sample [4]
...Collected sample [5]
...Collected sample [6]
...Collected sample [7]
...Collected sample [8]
...Collected sample [9]
...Collected sample [10]
...Collected sample [11]
...Collected sample [12]
...Collected sample [13]
...Collected sample [14]
...Collected sample [15]
...Collected sample [16]
...Collected sample [17]
...Collected sample [18]
...Collected sample [19]
...Collected sample [20]
Total samples collected: 35
--- SAMPLE COLLECTION ---
```

```
--- SAMPLE ANALYSIS ---
Starship has already landed at base: Charging docking station-X1A . Samples can be analysed while in flight for better analysis profitability.
--- SAMPLE ANALYSIS ---


--- LAUNCH ---
Initiating launch towards destination: Destination R035
Launching Starship towards destination: Destination R035
--- LAUNCH ---


--- FLY ---
Starship in flight towards base destination: Destination R035
--- FLY ---


--- LAND ---
Initiating Starship landing at Base: Charging docking station-X9C
--- LAND ---


--- LAUNCH ---
Initiating launch towards destination: Destination R478
Launching Starship towards destination: Destination R478
--- LAUNCH ---
```

```
--- FLY ---
Starship in flight towards base destination: Destination R478
--- FLY ---


--- LAND ---
Initiating Starship landing at Base: Charging docking station-X32
--- LAND ---


--- LAUNCH ---
Initiating launch towards destination: Destination R111
Launching Starship towards destination: Destination R111
--- LAUNCH ---


--- FLY ---
Starship in flight towards base destination: Destination R111
--- FLY ---


--- LAND ---
Initiating Starship landing at Base: Charging docking station-X22
--- LAND ---

Number of Flights done by Starship = 4

Process finished with exit code 0
```

5. DIP:

```
================================================================================================
DIP : Financial Payment Gateway scenario where Customer can payment through a Payment Agent
================================================================================================
*** Failing Dependency Inversion Principle
=================================================================
Transaction Scenario - 1 : Payment Transfer processed successfully
=================================================================
...Initiating Payment Transfer...
SBI initiated Wire transfer: Transaction ID: T-1, Amount: $800.0
Payment processed via SBI, Processed Amount: $800.0
Payment transaction ID: T-1, Amount: 800.0, Payment Status: Processed
Payment Status : Processed


...Initiating Payment Transfer...
SBI initiated Wire transfer: Transaction ID: T-2, Amount: $199.99
Payment processed via SBI, Processed Amount: $199.99
Payment transaction ID: T-2, Amount: 199.99, Payment Status: Processed
Payment Status : Processed


...Initiating Payment Transfer...
SBI initiated Wire transfer: Transaction ID: T-3, Amount: $49.49
Payment processed via SBI, Processed Amount: $49.49
Payment transaction ID: T-3, Amount: 49.49, Payment Status: Processed
Payment Status : Processed
```

```
============================================================
Transaction Scenario - 2 : Payment Transfer Refund scenario
============================================================
...Initiating Payment Transfer...
SBI initiated Wire transfer: Transaction ID: T-4, Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Initiated
Payment initiated, but not processed completely!
Payment Status : Pending


Payment processed via SBI, Processed Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Processed
Payment Status : Processed


Refund processed via SBI, Refunded Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Refunded
Payment Status : Refunded
================================================================
Transaction - 3 : Bank Manager to view Daily Transaction details
================================================================
Transaction History:

-- 1 --------------------------------------
ID        : T-1
Date Time : 2023-07-09T09:45:49.588819500
Amount    : 800.0
Remarks   : Rental amount transfer
Status    : Processed
-----------------------------------------
-- 2 --------------------------------------
ID        : T-2
Date Time : 2023-07-09T09:45:49.627716800
Amount    : 199.99
Remarks   : Toaster Oven Purchase
Status    : Processed
-----------------------------------------
-- 3 --------------------------------------
ID        : T-3
Date Time : 2023-07-09T09:45:49.627716800
Amount    : 49.49
Remarks   : Hiking Shoes Purchase
Status    : Processed
-----------------------------------------
-- 4 --------------------------------------
ID        : T-4
Date Time : 2023-07-09T09:45:49.627716800
Amount    : 350.0
Remarks   : Bose Headphone purchase
Status    : Refunded
-----------------------------------------
Total Money Spent with SBI: $1049.48
Average Customer Transfer with SBI: $349.82666666666665

Process finished with exit code 0
```

GOOD:

1. SRP:

```
=============================================================================================================
SRP : Quiz application scenario where quiz class is split across multipel clas having separate single responsibilities
=============================================================================================================
*** Adhering Single Responsibility Principle
Questions loaded successfully!
Question : If a, 4, b are in Arithmetic Progression; a, 2, b are in Geometric Progression; then a, 1, b are in:
Option 1 : H.P
Option 2 : A.P
Choose your option [1/2]:
1
Question : If a, b, c are in arithmetic progression, then:
Option 1 : 2b = a+c
Option 2 : b = a+c
Choose your option [1/2]:
1
Question : The sum of arithmetic progression 2, 5, 8, …, up to 50 terms is:
Option 1 : 3775
Option 2 : 3757
Choose your option [1/2]:
1
Question : Which of the following is an example of a geometric sequence?
Option 1 : 9, 20, 21, 28
Option 2 : 1, 2, 4, 8
Choose your option [1/2]:
2
Question : The next term of the given sequence 1, 5, 14, 30, 55, … is:
Option 1 : 91
Option 2 : 96
Choose your option [1/2]:
2
==== Quiz Completed Successfully! ====
Congratulations!
Quiz completed Successfully! Here are the results:
Total Questions    : 5
Correct Answers    : 3
In Correct Answers : 2
Percentage obtained: 60
Letter Grade       : C
```

```
==== Saving Quiz Result to file ====
[**SUCCESS**] Quiz Result saved successfully!
```

2. OCP:

```
================================================================================================================
OCP : Restaurant Billing Order scenario where for Festivals or Special offer class doesn't offer Extensibility
================================================================================================================
*** Adhering to Open Close Principle

======================================================================
Scenario: Special Offer Billing Order on a Festival (like Canada Day)
======================================================================
---------- INVOICE ----------
Order date    : Sun Jul 09 09:47:49 UYT 2023
Order ID      : BILL_1149

----------------------
Ordered Items :
----------------------
Chicken Shawarma Plate (Middle-Eastern Cuisine) ~~~~~~~ $16.79
Mixed Grill Plate (Middle-Eastern Cuisine) ~~~~~~~ $18.89
8oz Hummus (Middle-Eastern Cuisine) ~~~~~~~ $7.99
Baklava (Middle-Eastern Dessert) ~~~~~~~ $7.96
Coca-Cola (Cold Drink) ~~~~~~~ $5.98

Payment Method: Master Card
---------------------------
SubTotal      : $57.61
HST 15.000%   : $9.0
Total         : $66.2515
---------------------------
Cash Back Amount (25.0%)           : $16.0
Total payable Bill after Cash Back : $50.0
---------------------------

Process finished with exit code 0
```

3. LSP:

```
============================================================
LSP : Event Schedule generation for different type of events
============================================================
*** Partially Adhering to Liskov Substitution Principle

============================================================
Scenario 1:' Music Concert ' Event type
============================================================
Concert Schedule:
------------------------------
Name    : Dartmouth Summer Sunshine Free Concerts
Date    : 2023-07-08
------------------------------
Time    : 2:00 PM - 3:00 PM
Artist  : Maura Whitman
------------------------------
Time    : 3:00 PM - 4:00 PM
Artist  : Brooklyn Blackmore
------------------------------
```

```
============================================================
Scenario 2:' Exhibition ' Event type
============================================================
Exhibition Schedule:
-----------------------------
Name    : Annapolis Valley Exhibition
-----------------------------
Session : Scarecrow contest
Time    : 10:00 AM - 3:00 PM
Date    : 14 August, 2023
-----------------------------
Session : Scarecrow contest
Time    : 10:00 PM - 3:00 PM
Date    : 15 August, 2023
-----------------------------
Session : Riverside stage
Time    : 8:30 PM - 9:45 PM
Date    : 19 August, 2023
-----------------------------
```

```
============================================================
Scenario 3: ' Conference ' Event type
============================================================
Conference Schedule:
-----------------------------
Name    : Human Resource Management Conference
Date    : 2023-08-15
-----------------------------
Session : Registration and Welcome
Time    : 7:30 AM - 12:00 PM
Speaker : Chair person
-----------------------------
Session : Pre-conference session
Time    : 8:15 AM - 11:15 AM
Speaker : Chair person
-----------------------------
Session : Lunch Break
Time    : 1:30 PM - 2:00 PM
Speaker : Organising team
-----------------------------
Session : Socialising strategy
Time    : 2:00 PM - 3:15 PM
Speaker : Zen Sai
-----------------------------
Session : CLosing ceremony
Time    : 3:15 PM - 3:30 PM
Speaker : Chair person
-----------------------------

Process finished with exit code 0
```

4. ISP:

```
============================================================================================
ISP : SpaceX spacecraft resource exploration scenario
============================================================================================
*** Adhering Interface Segregation Principle

============================================================================================
SpaceX : Rover - Responsible for ONLY locating resource and NOT analysing resource
============================================================================================

--- FLY ---
Rover has already landed at base: Home Base and is Charging.
Possible options: Finalize the new location to search for possible resources, and launch Rover.

--- FINDING SAMPLE SOURCE ---
Rover is at base: Home Base and is Charging.
Rover Ready for launch at a new destination!
```

```
--- LAND ---
Rover docked at Base: Charging docking station-X22. Initiating charging.

--- FLY ---
Rover has already landed at base: Charging docking station-X22 and is Charging.
Possible options: Finalize the new location to search for possible resources, and launch Rover.

--- COMMUNICATION ---
...Establishing connection with Earth...
...Sending sample's analysis result report to Earth...

--- FINDING SAMPLE SOURCE ---
Rover is at base: Charging docking station-X22 and is Charging.
Rover has found minerals and resources at destination: Destination A312. Please send a SpaceX-Starship to analyse mineral resource samples at the destination.

================================================================================
SpaceX : Starship - Responsible for ONLY analysing resource and NOT locating resource
================================================================================

--- FLY ---
Starship has already landed at base: Home Base
Possible options: Check, Finalize operations, and initiate launch to another base destination.

--- LAUNCH ---
Initiating launch towards destination: Destination R998
Launching Starship towards destination: Destination R998

--- FLY ---
Starship in flight towards base destination: Destination R998
```

```
--- SAMPLE COLLECTION ---
Starship in flight. Initiating sample collection...
...Collected sample [1]
...Collected sample [2]
...Collected sample [3]
...Collected sample [4]
...Collected sample [5]
...Collected sample [6]
...Collected sample [7]
...Collected sample [8]
...Collected sample [9]
...Collected sample [10]
...Collected sample [11]
...Collected sample [12]
...Collected sample [13]
...Collected sample [14]
...Collected sample [15]
Total samples collected: 15

--- SAMPLE ANALYSIS ---
Starship has already landed at base: Home Base . Samples can be analysed while in flight for better analysis profitability.

--- LAND ---
Initiating Starship landing at Base: Charging docking station-X1A

--- FLY ---
Starship in flight towards base destination: Destination R998

--- COMMUNICATION ---
...Establishing connection with Earth...
...Sending sample's analysis result report to Earth...
```

```
--- SAMPLE COLLECTION ---
Starship in flight. Initiating sample collection...
...Collected sample [1]
...Collected sample [2]
...Collected sample [3]
...Collected sample [4]
...Collected sample [5]
...Collected sample [6]
...Collected sample [7]
...Collected sample [8]
...Collected sample [9]
...Collected sample [10]
...Collected sample [11]
...Collected sample [12]
...Collected sample [13]
...Collected sample [14]
...Collected sample [15]
...Collected sample [16]
...Collected sample [17]
...Collected sample [18]
...Collected sample [19]
...Collected sample [20]
Total samples collected: 35

--- SAMPLE ANALYSIS ---
Starship has already landed at base: Charging docking station-X1A . Samples can be analysed while in flight for better analysis profitability.

--- LAUNCH ---
Initiating launch towards destination: Destination R035
Launching Starship towards destination: Destination R035

--- FLY ---
Starship in flight towards base destination: Destination R035

--- LAND ---
Initiating Starship landing at Base: Charging docking station-X9C
```

```
--- LAUNCH ---
Initiating launch towards destination: Destination R478
Launching Starship towards destination: Destination R478

--- FLY ---
Starship in flight towards base destination: Destination R478

--- LAND ---
Initiating Starship landing at Base: Charging docking station-X32

--- LAUNCH ---
Initiating launch towards destination: Destination R111
Launching Starship towards destination: Destination R111

--- FLY ---
Starship in flight towards base destination: Destination R111

--- LAND ---
Initiating Starship landing at Base: Charging docking station-X22
Number of Flights done by Starship = 4

Process finished with exit code 0
```

5. DIP:

```
================================================================================================
DIP : Financial Payment Gateway scenario where Customer can payment through a Payment Agent
================================================================================================
*** Adhering to Dependency Inversion Principle

=====================================================================
Transaction Scenario - 1 : Payment Transfer processed successfully
=====================================================================
...Initiating Payment Transfer...
SBI initiated Money transfer: Transaction ID: T-1, Amount: $800.0
Payment processed via SBI, Processed Amount: $800.0
Payment transaction ID: T-1, Amount: 800.0, Payment Status: Processed
Payment Status : Processed


...Initiating Payment Transfer...
SBI initiated Money transfer: Transaction ID: T-2, Amount: $199.99
Payment processed via SBI, Processed Amount: $199.99
Payment transaction ID: T-2, Amount: 199.99, Payment Status: Processed
Payment Status : Processed


...Initiating Payment Transfer...
SBI initiated Money transfer: Transaction ID: T-3, Amount: $49.49
Payment processed via SBI, Processed Amount: $49.49
Payment transaction ID: T-3, Amount: 49.49, Payment Status: Processed
Payment Status : Processed
```

```
=========================================================
Transaction Scenario - 2 : Payment Transfer Refund scenario
=========================================================
...Initiating Payment Transfer...
SBI initiated Money transfer: Transaction ID: T-4, Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Initiated
Payment initiated, but not processed completely!
Payment Status : Initiated


Payment processed via SBI, Processed Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Processed
Payment Status : Processed


Refund processed via SBI, Refunded Amount: $350.0
Payment transaction ID: T-4, Amount: 350.0, Payment Status: Refunded
Payment Status : Refunded
```

```
================================================================
Transaction - 3 : Bank Manager to view Daily Transaction details
================================================================

Transaction History:

-- 1 -------------------------------------
ID        : T-1
Date Time : 2023-07-09T09:49:22.511270200
Amount    : 800.0
Remarks   : Rental amount transfer
Status    : Processed
-----------------------------------------
-- 2 -------------------------------------
ID        : T-2
Date Time : 2023-07-09T09:49:22.542187500
Amount    : 199.99
Remarks   : Toaster Oven Purchase
Status    : Processed
-----------------------------------------
-- 3 -------------------------------------
ID        : T-3
Date Time : 2023-07-09T09:49:22.542187500
Amount    : 49.49
Remarks   : Hiking Shoes Purchase
Status    : Processed
-----------------------------------------
-- 4 -------------------------------------
ID        : T-4
Date Time : 2023-07-09T09:49:22.542187500
Amount    : 350.0
Remarks   : Bose Headphone purchase
Status    : Refunded
-----------------------------------------
Total Money Spent with SBI: $1049.48
Average Customer Transfer with SBI: $349.82666666666665

Process finished with exit code 0
```