

Assignment Report for Assignment-05

Course and Section	CSC.215
Assignment Name	Assignment-05
Due Date and Time	4/22/2024 @ 11:59 PM
First Name and Last Name	Marty Martin
SFSU Email Account	pmartin@sfsu.edu
First Name and Last Name of Teammate	
SFSU Email Account of Teammate	

**PART # A****Question Description and Analysis:**

Please choose 1 guideline and explain all its bullet point(s) in detail.

- Use at least one-half of a page to explain the guideline.
- Then write a code example to demonstrate your application of the guideline in designing a class.
- Then explain in at least one-half of a page where and how you applied the guideline in the code.

Answer:

Consistency is a crucial aspect of object-oriented programming that ensures code is easy to understand, maintain and navigate. It involves adhering to established coding conventions and styles, such as using consistent naming for variables, methods, and classes, and structuring code blocks in a logical order. For instance, constructors should be placed before methods. This approach helps developers quickly comprehend new codebases and predict the location and behavior of various components, reducing cognitive load and minimizing errors. Consistency also involves applying design patterns and architectural principles, which ensures that similar problems are solved using similar methods throughout the software. This approach enhances the coherence and quality of the code.

```
package utilities;

public class Time {
    private int hours;
    private int minutes;
    private int seconds;

    public Time() {
        this(0, 0, 0);
    }

    // Constructor with all parameters
    public Time(int hours, int minutes, int seconds) {
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    // Consistent naming for getters
    public int getHours() {
        return hours;
    }

    public int getMinutes() {
        return minutes;
    }

    public int getSeconds() {
        return seconds;
    }

    // Consistent naming for setters
    public void setHours(int hours) {
        this.hours = hours;
    }

    public void setMinutes(int minutes) {
        this.minutes = minutes;
    }
}
```

```
public void setSeconds(int seconds) {  
    this.seconds = seconds;  
}  
  
// Example of a consistently named utility method  
public void resetToMidnight() {  
    hours = 0;  
    minutes = 0;  
    seconds = 0;  
}  
}
```

Consistency is an essential aspect when designing a Time class. It enhances readability and maintainability, making the code more approachable and understandable. By following Java's standard naming conventions, like using camelCase for methods (getHours, setMinutes, and resetToMidnight) and capitalizing class names, the code becomes more organized and uniform. Organizing constructors above methods within the class structure allows developers, whether familiar with the specific project or not, to quickly locate and understand the purpose and initialization process of class instances. The class also maintains a uniform approach in method functionality, with getters for retrieving values and setters for modifying values. This ensures that interactions are predictable and that the object's state is always controlled and traceable. Consistent coding practices simplify the debugging and extension of the class and ensure that it can be easily integrated and utilized by other software parts without unexpected behaviors.

Screenshots of Outputs and Explanation:

Nothing Follows...

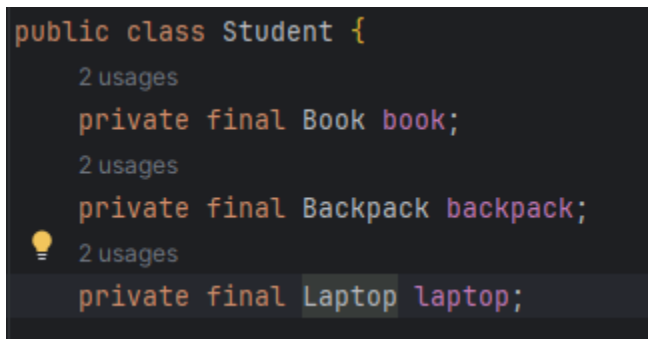
PART # B**Question Description and Analysis:**

Create a program to demonstrate your understanding of the class relationship HAS-A.

- - Your program must consist of at least 5 classes:
- - 1 (user-defined) driver class that contains the main method (and no constructors or only private constructors)
- - 4 user-defined data type classes: 1 data type should have data fields (properties) of the 3 other user-defined data types. For example, Student HAS-A 1 Book property, 1 Backpack property, and 1 Laptop property.
- - Your program must be meaningfully organized in at least 5 Java (.java extension) files.
- - The main method in the driver class must consist of only 1 statement which is a method call.
- - Generate or draw a (simple) diagram of the class relationships among your program's classes.
- - The program's design and code must demonstrate your understanding of the class relationship HAS-A.
- - The program's output must demonstrate your proper use of the class relationship HAS-A in your program's design and code.

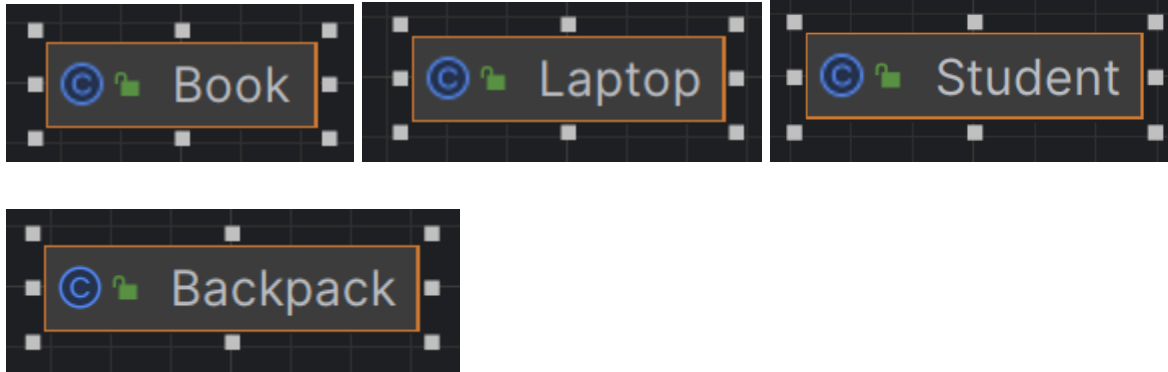
Answer:

This is my answer...



```
public class Student {  
    2 usages  
    private final Book book;  
    2 usages  
    private final Backpack backpack;  
    2 usages  
    private final Laptop laptop;
```

Screenshots of Outputs and Explanation:



PART # B-1**Question Description and Analysis:**

In one-third of a page or more, explain in detail your understanding of the problem. What is this question asking you to do? What does the client want? What are the requirements of the program? What are the important details you observed from the provided information? How do you plan to solve the problem? How do you organize/design the solution? What are the important elements of your program? What will you pay attention to when you code the solution? Think about Interviews and Problem-Solving.

Answer:

In Part B, you need to use the "HAS-A" relationship in object-oriented programming to solve a problem. This means one class contains one or more instances of another class. Create a program with a driver class and four user-defined classes, representing different entities. Analyze the client's requirements, ensure successful program compilation, and demonstrate the HAS-A relationship through logical interactions. Organize the program across multiple Java files, implement constructors and methods, and focus on encapsulation and access control. Following these guidelines will showcase the HAS-A relationship, meet the client's expectations, and provide a foundation for enhancements.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished...

PART # B-3**Question Description and Analysis:**

In one-third of a page or more, analyze your results and plan for future development. Does your program compile? Does it satisfy all the client's requirements? What works and why? What does not work and why? What can be improved and how? What is your plan to improve the program in the future? Think about Interviews

Answer:

Part B requires creating a program that demonstrates the "HAS-A" relationship in object-oriented programming. You need to use one class that contains one or more instances of another class. To achieve this, you need to create four user-defined classes and analyze the client's requirements, ensure successful program compilation and demonstrate the HAS-A relationship through logical interactions. The program should be organized across multiple Java files, implement constructors and methods, focus on encapsulation and access control, and meet the client's expectations.

To plan for future development, you need to ensure the program compiles without errors, meets all specified requirements, and demonstrates the HAS-A relationship effectively. Future development should focus on enhancing the program's robustness and scalability by adding more complex interactions between objects, introducing error handling, and refactoring the existing code. You should aim to make the application more user-centric by developing a user interface that allows dynamic interaction with the object model.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished...

PART # C**Question Description and Analysis:**

Create a program to demonstrate your understanding of class relationship Inheritance IS-A.

- - Your program must consist of at least 4 classes:
 - - 1 (user-defined) driver class that contains the main method (and no constructors or only private constructors)
 - - 3 user-defined data type classes that are related by the 3-level (3-layer) class relationship Inheritance IS-A.
- For example, ComputerScienceStudent IS-A Student, and Student IS-A Person.
 - - Your program must be meaningfully organized in at least 4 Java (.java extension) files.
 - - The main method in the driver class must consist of only 1 statement which is a method call.
 - - Generate or draw a (simple) diagram of the class relationships among your program's classes.
 - - The program's design and code must demonstrate your understanding of the class relationship Inheritance IS-A.
 - - The program's output must demonstrate your proper use of the class relationship Inheritance IS-A in your program's design and code.

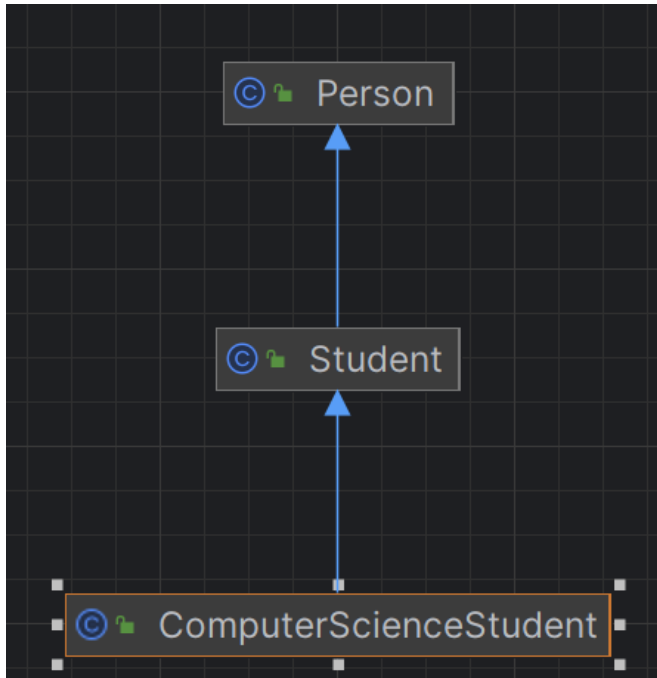
Answer:

This is my answer...

```
2 usages
public class ComputerScienceStudent extends Student {
    3 usages
    private String favoriteProgrammingLanguage;

    1 usage
    public ComputerScienceStudent(String name, int age, String studentId, String favoriteProgrammingLanguage) {
        super(name, age, studentId);
        this.favoriteProgrammingLanguage = favoriteProgrammingLanguage;
    }
}
```

Screenshots of Outputs and Explanation:



PART # C-1

Question Description and Analysis:

In one-third of a page or more, explain in detail your understanding of the problem. What is this question asking you to do? What does the client want? What are the requirements of the program? What are the important details you observed from the provided information? How do you plan to solve the problem? How do you organize/design the solution? What are the important elements of your program? What will you pay attention to when you code the solution? Think about Interviews and Problem-Solving.

Answer:

Part C requires a program that models a three-tiered class hierarchy: **Person**, **Student** (subclass of **Person**), and **ComputerScienceStudent** (subclass of **Student**). Each class must inherit properties and methods from its predecessor while introducing any unique characteristics. The assignment includes designing each class to ensure proper inheritance, handling overridden methods, and organizing the code across at least four Java files. The driver class should instantiate and utilize

objects from these classes with minimal logic. This showcases clear, maintainable code that underscores the power and utility of inheritance in software development.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished...

PART # C-3

Question Description and Analysis:

In one-third of a page or more, analyze your results and plan for future development. Does your program compile? Does it satisfy all the client's requirements? What works and why? What does not work and why? What can be improved and how? What is your plan to improve the program in the future? Think about Interviews

Answer:

To analyze the results and plan for future development of the Inheritance (IS-A) relationship program, ensure it compiles and runs without errors, accurately demonstrating hierarchical relationships among the classes. Each class should exhibit accurate inherited properties and behaviors, while subclasses should extend or modify their superclass traits appropriately. Potential improvements include introducing additional subclasses or features, refining the program to incorporate more complex behaviors, or enhancing user interaction. Plans for future development should increase the robustness and scalability of the application by integrating more advanced object-oriented features.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished...

