THE UNIVERSITY OF SYDNEY

SOFT2201/COMP9201 Tutorial 5

Creational Design Patterns

Creational Design Patterns

Creational design patterns handle object instantiation and construction mechanisms. Being able to manage instantiation of multiples of types, abstracting construction and process.

Question 1: Issues with constructors

Assignment Project Exam Help

Identify some potential issues with the code segment below. Consider the following criteria: Readability, Maintainability, Dependencies and Reusability.

https://tutorcs.com

```
public class VehicleApplication {
    private static void addVehicle(int numPassengers,
                                    int numWheels,
                                    String colour) {
        if ((numPassengers == 4 || numPassengers == 5)
                    && numWheels == 4) {
            vehicles.add(new Car(numPassengers, colour));
        } else if (numPassengers == 2 && numWheels == 4) {
            vehicles.add(new Ute(colour));
        } else if (numPassengers == 2 && numWheels == 2) {
            vehicles.add(new Motorbike(colour));
        } else {
            System.out.println("Invalid input");
        }
    }
}
```

WeChat: cstutorcs

Question 2: Is this better?

As a class, identify problems within the following code. Consider aspects where we may need to extend the code or refactor it.

```
enum VehicleType {
    CAR4,
    CAR5,
    MOTORBIKE,
    UTE
}

public class VehicleFactory {
    public static Vehicle make(VehicleType type, String colour) {
        switch (type) {
            case: VehicleType.CAR4:
                return new Car(4, colour);
            case: VehicleType.CAR5:
                return new Car(5, colour);
            case: VehicleType.ElectricCar4:
                return new ElectricCar(4, colour);
            case: VehicleType.ElectricCar5:
                return new ElectricCar(5, colour);
            case: VehicleType.MOTORBIKE:
                return new Motorbike(colour);
            case: VehicleType.Scooter:
                return new Scooter(colour);
            case: VehicleType.UTE:
                return new Ute(colour);
        }
        return null;
    }
}

//continued
```

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

```java
public class VehicleApplication {
    private static void addVehicle(int numPassengers,
                                   int numWheels,
                                   String colour) {
        if (numPassengers == 4 && numWheels == 4) {
            vehicles.add(VehicleFactory.make(VehicleType.CAR4,
                                             colour);
        else if (numPassengers == 5 && numWheels == 4) {
            vehicles.add(VehicleFactory.make(VehicleType.CAR5,
                                                         colour);
        } else if (numPassengers == 2 && numWheels == 4) {
            vehicles.add(VehicleFactory.make(VehicleType.UTE,
                                                         colour);
        } else if (numPassengers == 2 && numWheels == 2) {
            vehicles.add(VehicleFactory.make(VehicleType.MOTORBIKE,
                                                         colour);
        } else {
            System.out.println("Invalid input");
        }
    }
}
```

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Factory method

Factory Method specifies construction of multiple related objects under a single method, the object can decide what object to create and how to create it and return to the caller. The object created is utilised through a common type that all objects created from the factory method type share.

Question 3: Refactor again

Using your knowledge of factory method, map out the previous code (`VehicleFactory`) and refactor the design to utilise factory method pattern.

- How would you break up the invocation of objects?
- How would the programmer invoke an object of this type?

Builder

The builder pattern creates an abstraction of the construction process of an object, particularly complex objects with multiple constructors. This allows for the object to maintain a single construction method and the builder object to manage the construction process.

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Question 4: Discuss object construction strategy

As a class, discuss how you can transform the following code to use the `Builder` pattern and how a programmer will utilise this object.

```java
public class Computer {
    private Part cpu;
    private Part motherboard;
    private ArrayList<Part> hdds;
    private ArrayList<Part> ram;
    private Enclosure encl;

    public Computer(CPU cpu,
                    Motherboard motherboard) {
        this.cpu = cpu;
        this.motherboard = motherboard;
        hdds = new ArrayList<>();
        ram = new ArrayList<>();
        encl = null;
    }

    public Computer(CPU cpu,
                    Motherboard motherboard,
                    Enclosure enclosure) {
        this(cpu, motherboard);
        encl = enclosure;
    }

    public Computer(CPU cpu,
                    Motherboard motherboard,
                    Enclosure enclosure,
                    ArrayList<Part> hdds) {
        this(cpu, motherboard, enclosure);
        this.hdds = hdds;
    }

    public Computer(CPU cpu,
                    Motherboard motherboard,
                    Enclosure enclosure,
                    ArrayList<Part> hdds,
                    ArrayList<Part> ram) {
        this(cpu, motherboard, enclosure, hdds);
        this.ram = ram;
    }
    /* Getters and Setters */
}
```

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

- How would a programmer construct this object?
- How many construction variations exist? Is there a chance that it can be ambiguous?
- What kind of issues can you see with the list of constructors?
- Create a concrete builder type that will abstract the construction process of the object
- We want to now separate the Computer type into two separations

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs