Programming and design patterns Lab

Ex-11

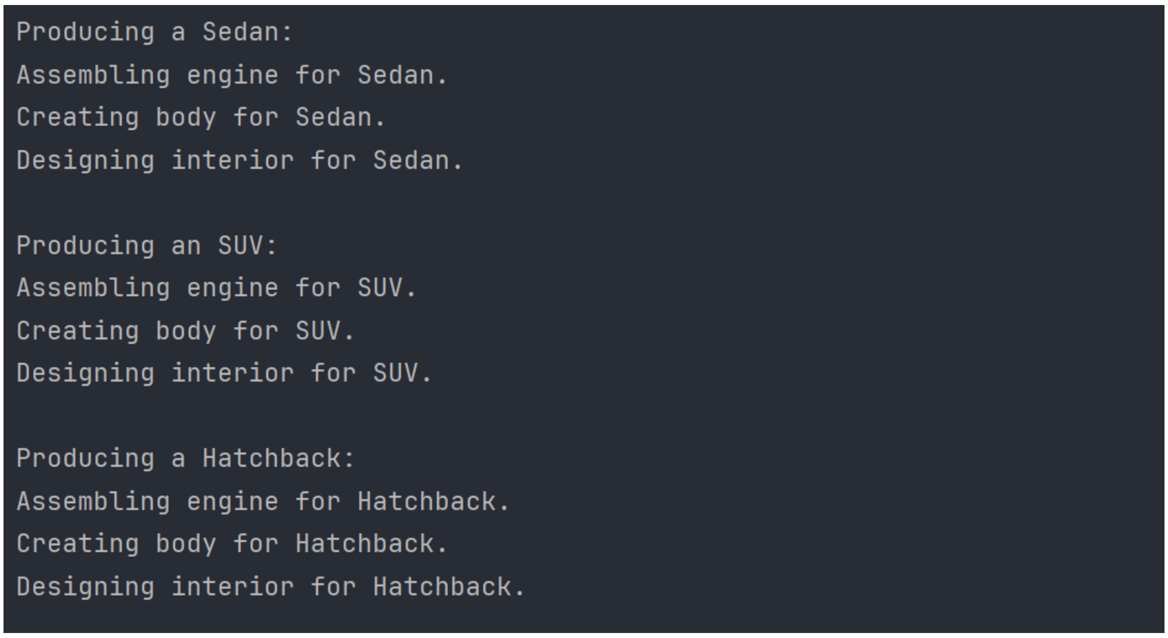
Mohit R
3122235002072
IT-b

CODE:

```
from abc import ABC, abstractmethod


class Engine(ABC):
    @abstractmethod
    def assemble(self):
        pass

class SedanEngine(Engine):
    def assemble(self):
        return "Assembling engine for Sedan."

class SUVEngine(Engine):
    def assemble(self):
        return "Assembling engine for SUV."

class HatchbackEngine(Engine):
    def assemble(self):
        return "Assembling engine for Hatchback."

# Body interface
class Body(ABC):
    @abstractmethod
    def create(self):
        pass

class SedanBody(Body):
    def create(self):
        return "Creating body for Sedan."

class SUVBody(Body):
    def create(self):
        return "Creating body for SUV."

class HatchbackBody(Body):
    def create(self):
        return "Creating body for Hatchback."

# Interior interface
class Interior(ABC):
    @abstractmethod
    def design(self):
        pass

class SedanInterior(Interior):
    def design(self):
        return "Designing interior for Sedan."

class SUVInterior(Interior):
    def design(self):
        return "Designing interior for SUV."

class HatchbackInterior(Interior):
    def design(self):
        return "Designing interior for Hatchback."
```

```
class CarFactory(ABC):
    @abstractmethod
    def create_engine(self):
        pass

    @abstractmethod
    def create_body(self):
        pass

    @abstractmethod
    def create_interior(self):
        pass

class SedanFactory(CarFactory):
    def create_engine(self):
        return SedanEngine()

    def create_body(self):
        return SedanBody()

    def create_interior(self):
        return SedanInterior()

class SUVFactory(CarFactory):
    def create_engine(self):
        return SUVEngine()
    def create_body(self):
        return SUVBody()

    def create_interior(self):
        return SUVInterior()

class HatchbackFactory(CarFactory):
    def create_engine(self):
        return HatchbackEngine()

    def create_body(self):
        return HatchbackBody()

    def create_interior(self):
        return HatchbackInterior()


class CarProduction:
    def __init__(self, car_factory):
        self.engine = car_factory.create_engine()
        self.body = car_factory.create_body()
        self.interior = car_factory.create_interior()

    def produce_car(self):
        try:
            print(self.engine.assemble())
            print(self.body.create())
            print(self.interior.design())
        except Exception as e:
            raise RuntimeError(f"Error in car production: {e}")


def main():
    sedan_factory = SedanFactory()
    sedan_production = CarProduction(sedan_factory)
    print("Producing a Sedan:")
    sedan_production.produce_car()
    suv_factory = SUVFactory()
    suv_production = CarProduction(suv_factory)
    print("\nProducing an SUV:")
    suv_production.produce_car()
    hatchback_factory = HatchbackFactory()
    hatchback_production =
CarProduction(hatchback_factory)
    print("\nProducing a Hatchback:")
    hatchback_production.produce_car()

if __name__ == "__main__":
    main()
```

OUTPUT:



```
Producing a Sedan:
Assembling engine for Sedan.
Creating body for Sedan.
Designing interior for Sedan.

Producing an SUV:
Assembling engine for SUV.
Creating body for SUV.
Designing interior for SUV.

Producing a Hatchback:
Assembling engine for Hatchback.
Creating body for Hatchback.
Designing interior for Hatchback.
```