Template Method Pattern

Introduction

- Say that XYZ provides two high-tech programs: computer science and electrical engineering. Students must complete the study plan to obtain their certificate. Both programs have their own study plans that need a sequence of courses. Let's take a look at this program.
- We're going to start with this case study:

    - Say XYZ school offers two high tech programs: Computer Science, and Electrical Engineering. Students will need to follow the study plan to get their certificates. These two programs have their own study plans that require a sequence of courses. Let's look at the following program.

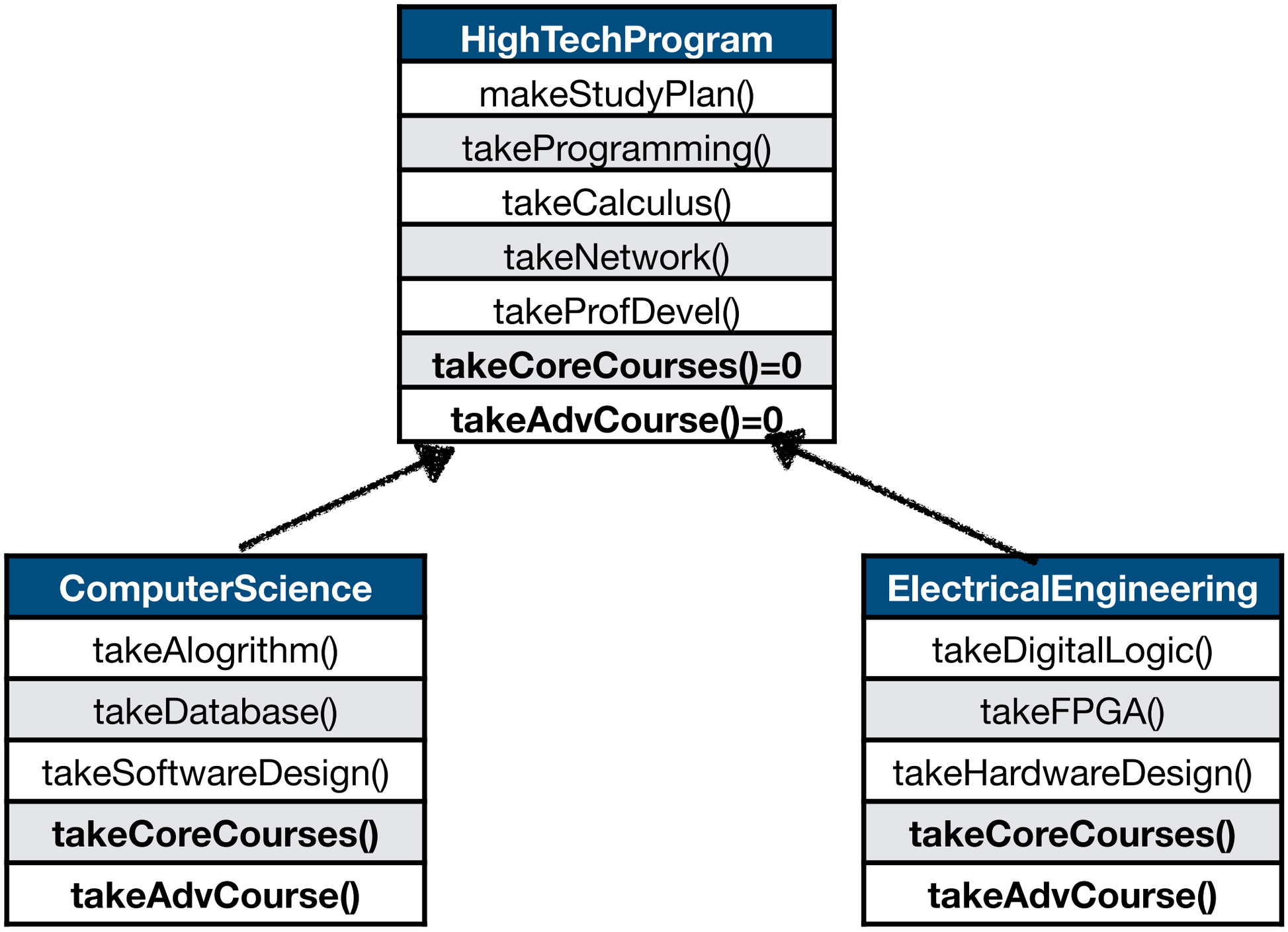
Example of Study Plan

```
class ComputerScience:
    def makeStudyPlan(self):
        self.takeProgramming()
        self.takeCalculus()
        self.takeAlogrithm()
        self.takeDatabase()
        self.takeNetwork()
        self.takeProfDevel()
        self.takeSoftwareDesign()

    def takeProgramming(self):
        print("Taking C and C++")

    def takeCalculus(self):
        print("Taking Calculus I & II")

    def takeAlogrithm(self):
        print("Taking Data Structures and Discrere Math")

    def takeDatabase(self):
        print("Taking Database Design")

    def takeNetwork(self):
        print("Taking Computer Networks")

    def takeProfDevel(self):
        print("Taking Professional Development")

    def takeSoftwareDesign(self):
        print("Taking Software Design")
```

```
class ElectricalEngineering:
    def makeStudyPlan(self):
        self.takeProgramming()
        self.takeCalculus()
        self.takeDigitalLogic()
        self.takeFPGA()
        self.takeNetwork()
        self.takeProfDevel()
        self.takeHardwareDesign()

    def takeProgramming(self):
        print("Taking C and C++")

    def takeCalculus(self):
        print("Taking Calculus I & II")

    def takeDigitalLogic(self):
        print("Taking Digital Logic and Circuit Design")

    def takeFPGA(self):
        print("Taking FPGA Design")

    def takeNetwork(self):
        print("Taking Computer Networks")

    def takeProfDevel(self):
        print("Taking Professional Development")

    def takeHardwareDesign(self):
        print("Taking Hardware Design")
```

Improved Version

- Since the study plans of these two programs are very similar except few different courses. So we can create a template for the study plan for these programs to follow.



Improved Version

```
class HighTechProgram(ABC):
    def makeStudyPlan(self):
        self.takeProgramming()
        self.takeCalculus()
        self.takeCoreCourses()
        self.takeNetwork()
        self.takeProfDevel()
        self.takeAdvCourse()

@abstractmethod
    def takeCoreCourses(self):
        pass

    @abstractmethod
    def takeAdvCourse(self):
        pass
```

```
class ComputerScience(HighTechProgram):
    def takeCoreCourses(self):
        self.takeAlogrithm()
        self.takeDatabase()

    def takeAdvCourse(self):
        self.takeSoftwareDesign()
```

```
class ElectricalEngineering(HighTechProgram):
    def takeCoreCourses(self):
        self.takeDigitalLogic()
        self.takeFPGA()

    def takeAdvCourse(self):
        self.takeHardwareDesign()
```

The Template Method Pattern

- It defines the skeleton of an algorithm in a method, by deferring certain steps to subclass. The Template method allows subclasses to redefine some steps in an algorithm without changing the algorithm's structure.

- To create a template method, you define an algorithm as a set of steps. One or more of these steps is defined to be abstract and implemented by a subclass. This ensures the algorithm structure is not modified.

A Hook

- A hook mark is a method defined in the abstract class, but only given as an empty or default implementation. This provides a subclass with an option to connect to the algorithm at different points if it so wishes. Or the sub-class can just ignore it because the hook is not an abstract method.

- For example, we can enhance our training program by asking a student if they want to take a practical project to their study plan or not.

Improved Version

```
class HighTechProgram(ABC):
    @final
    def makeStudyPlan(self):
        self.takeProgramming()
        self.takeCalculus()
        self.takeCoreCourses()
        self.takeNetwork()
        self.takeProfDevel()
        self.takeAdvCourse()
        if (self.wantProject()):
            self.takeProject();

    @abstractmethod
    def takeCoreCourses(self):
        pass

    @abstractmethod
    def takeAdvCourse(self):
        pass

    def wantProject(self):
        return False
```

```
class ComputerScience(HighTechProgram):
    def takeCoreCourses(self):
        self.takeAlogrithm()
        self.takeDatabase()

    def takeAdvCourse(self):
        self.takeSoftwareDesign()

    def wantProject(self):
        answer = input("Would you like to take Software Project (y/n): ")
        if answer == "y":
            return True
        else:
            return False
```

```
class ElectricalEngineering(HighTechProgram):
    def takeCoreCourses(self):
        self.takeDigitalLogic()
        self.takeFPGA()

    def takeAdvCourse(self):
        self.takeHardwareDesign()
```

Case Study - Car Sales Reporter

- In this example, sales records will be stored in a SQLite database table. SQLite is a simple file-based database engine that lets you store records using SQL syntax.
- In car sales reporter, there are two tasks we need to accomplish.:
  - Select all sales of new vehicles and display them on the screen in a comma format.
  - Generate a comma-delimited list of all salespeople with their gross sales and save in a file that can be imported into a spreadsheet.
- In either case, you will need the following steps:
  - Connect to the database.
  - Build a query for new cars or raw sales.
  - Issue the query.
  - Format results in a comma-delimited character string.
  - Send the data into one file.

This example comes from the “Python 3 Object-oriented Programming - Second Edition” textbook.

Case Study - Car Sales Reporter

Abstract Base Class

```
class QueryTemplate(ABC):
    def connect(self):
        self.conn = sqlite3.connect("sales.db")

    @abstractmethod
    def construct_query(self):
        pass

    def do_query(self):
        results = self.conn.execute(self.query)
        self.results = results.fetchall()

    def format_results(self):
        output = []
        for row in self.results:
            row = [str(i) for i in row]
            output.append(", ".join(row))
        self.formatted_results = "\n".join(output)

    @abstractmethod
    def output_results(self):
        pass

    def process_format(self):
        self.connect()
        self.construct_query()
        self.do_query()
        self.format_results()
        self.output_results()
```

Concrete Subclass

```
class NewVehiclesQuery(QueryTemplate):
    def construct_query(self):
        self.query = "select * from Sales where new='true'"

    def output_results(self):
        print(self.formatted_results)
```

Concrete Subclass

```
class UserGrossQuery(QueryTemplate):
    def construct_query(self):
        self.query = (
            "select salesperson, sum(amt) "
            + " from Sales group by salesperson"
        )

    def output_results(self):
        filename = "gross_sales_{0}".format(
            datetime.date.today().strftime("%Y%m%d")
        )
        with open(filename, "w") as outfile:
            outfile.write(self.formatted_results)
```

Hollywood Principle

- The Hollywood principle is a software design methodology: "Don't call us, we'll call you".
- The principle is a helpful paradigm that helps in code development with high cohesion and loose coupling that is easier to debug, maintain and test.
    - High cohesion consists in keeping your classes true to their intent and not allowing their behavior to extend beyond its primary objective.
    - Loose coupling involves reducing coupling within a software system. You can keep the classes loosely coupled to make sure you don't have unnecessary references.
- The majority of beginners are first introduced to programming from a directly opposite perspective. Programs like Hello World take control of the operating environment and rely on the underlying system for their work. On the other hand, linear flow programs are usually easy to understand.
- However, as systems become more complicated, the linear model becomes less easily maintained. If we leave the low-level components, to control the operations of the high-level components. The system may cause a dead end or incorrect sequences when multiple low-level components are running at the same time.
- By following the Hollywood Principal, we will let the top-level component decide when it calls the bottom-level components.

Summary

- A template method defines the steps of the algorithm, leaving it up to the subclasses to implement these steps.
- The abstract class of the template method can define concrete methods, abstract methods and hooks.
- The abstract methods are implemented by subclasses
- Hooks are just methods that don't do anything, or that execute a default behavior defined by the abstract class. But they can be overridden by the subclasses.
- To avoid the subclass overriding the template method, it must be defined as final.
- The Hollywood Principal guides us to the top-level components, to make decisions about how and when to call the bottom-level components.