

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 4**

з дисципліни “Програмування. Об'єктно-орієнтоване

програмування та шаблони проєктування”

тема “Породжувальні шаблони.”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-02  Красношапка Анастасія Андріївна  Варіант 5 |  | Перевірила  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна |

Київ 2021

**Мета роботи**

Ознайомитися зі породжувальними шаблонами, навчитися визначати, де саме потрібно використовувати кожен з розглянутих варінтів, а також розробляти програмне забезпечення на мові C# з їх використанням.

**Постановка задачі**

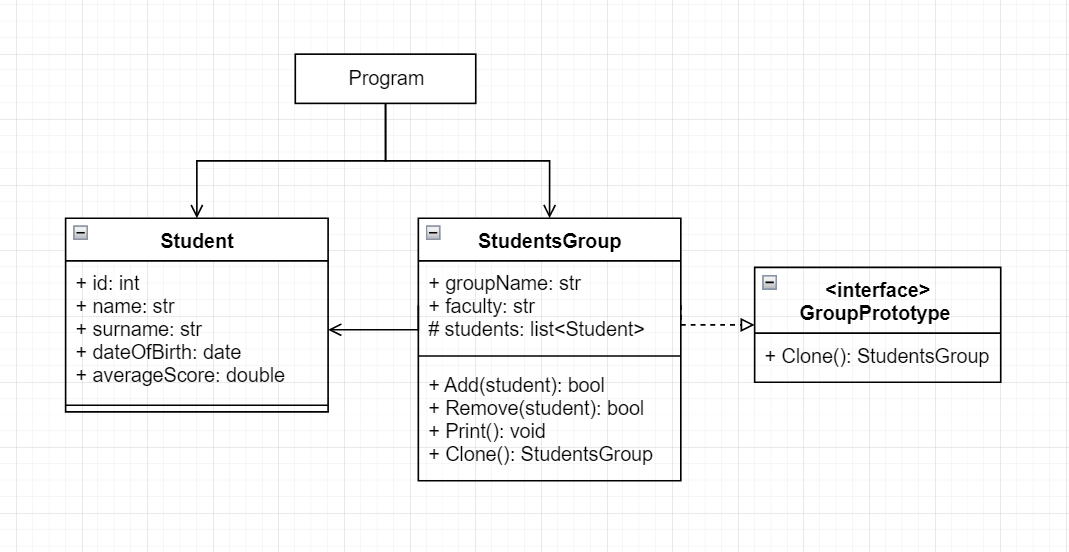
1. За допомогою шаблону проєктування розробити механізм клонування групи студентів, яка складається з окремих об’єктів класу «Студент». Дане клонування необхідне для обробки масиву даних про студентів різними службами університету: поліклінікою, деканатом, студентською радою і т.д.
2. За допомогою шаблону проєктування реалізувати процес випуску автомобілів різної комплектації. Так на різних етапах створення автівки на неї можна встановити двигуни різної потужності, різні типи кузова та по-різному оформити салон.

**Обгрунтування вибору шаблонів**

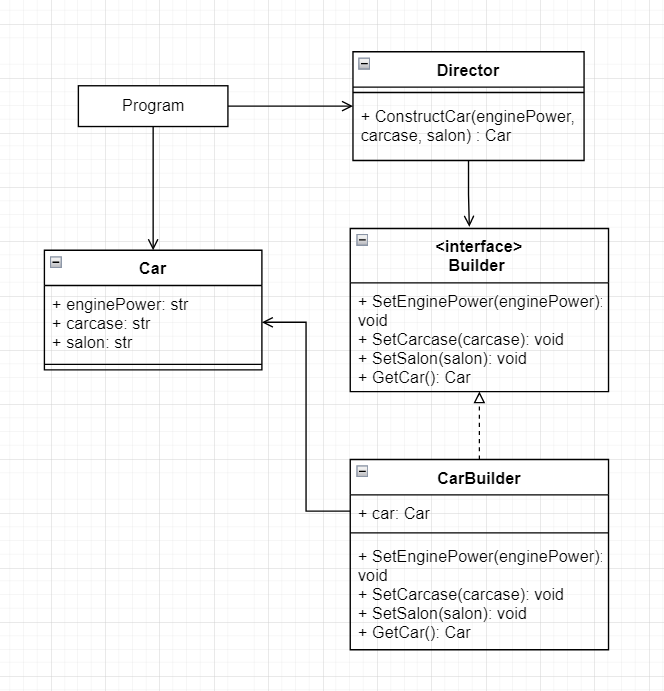
1. Для першого завдання був вибраний шаблон прототип (Prototype), оскільки цей шаблон дозволяє зробити точну копію об’єкта, не вдаючись в подробиці його реалізації.
2. Для другого завдання був обраний шаблон будівельник (Builder), оскільки він дозволяє створювати складні об’єкти крок за кроком, що саме й було потрібно в завданні при створенні автомобіля.

**UML діаграми класів**

Завдання 1



Завдання 2



**Текст програми**

Завдання 1

|  |
| --- |
| **Program.cs** |
| using System;  namespace task3  {      class Program      {          static void Main(string[] args)          {              StudentsGroup kp02 = new StudentsGroup("KP-02", "Faculty of Applied Mathematics");              kp02.Add(new Student(1, "Ana", "Davidson", new DateTime(2000,12,1), 95));              kp02.Add(new Student(2, "Mary", "Smith", new DateTime(2000,12,20), 90));              kp02.Add(new Student(3, "John", "Kane", new DateTime(2001,12,1), 92));              kp02.Add(new Student(4, "Mary", "Jane", new DateTime(2000,9,2), 87));              kp02.Add(new Student(5, "David", "Smith", new DateTime(2001,12,19), 95));              kp02.Add(new Student(6, "June", "Akana", new DateTime(2000,12,2), 90));              System.Console.WriteLine("Original group:");              kp02.Print();              // Get clone of group "KP-02" for policlinic, dean office or student council              StudentsGroup kp02Clone = kp02.Clone();              // Checking the clone              System.Console.WriteLine("\r\nClone group:");              kp02Clone.Print();          }      }  } |

|  |
| --- |
| **StudentsGroup.cs** |
| using System.Collections.Generic;  namespace task3  {      public class StudentsGroup: GroupPrototype      {          public string groupName;          public string faculty;          protected List<Student> students;          public StudentsGroup(string groupName, string faculty)          {              this.groupName = groupName;              this.faculty = faculty;              this.students = new List<Student>();          }          protected StudentsGroup(StudentsGroup group)          {              this.groupName = group.groupName;              this.faculty = group.faculty;              this.students = group.students;          }          public bool Add(Student student)          {              foreach(Student st in this.students)              {                  if(st.id == student.id)                  {                      return false;                  }              }              students.Add(student);              return true;          }          public bool Remove(Student student)          {              return students.Remove(student);          }          public void Print()          {              System.Console.WriteLine($"Group name: {this.groupName}\r\nFaculty: {this.faculty}\r\nStudents:");              foreach(Student student in this.students)              {                  System.Console.WriteLine(student);              }          }          public StudentsGroup Clone()          {              return new StudentsGroup(this);          }      }  } |

|  |
| --- |
| **Student.cs** |
| using System;  namespace task3  {      public class Student      {          public int id;          public string name;          public string surname;          public DateTime dateOfBirth;          public double averageScore;          public Student(int id, string name, string surname, DateTime dateOfBirth, double averageScore)          {              this.id = id;              this.name = name;              this.surname = surname;              this.dateOfBirth = dateOfBirth;              this.averageScore = averageScore;          }          public override string ToString()          {              return $"[{this.id}] {this.name} {this.surname} -- birthdate: {this.dateOfBirth.ToString("MM/dd/yyyy")}, average score: {this.averageScore};";          }      }  } |

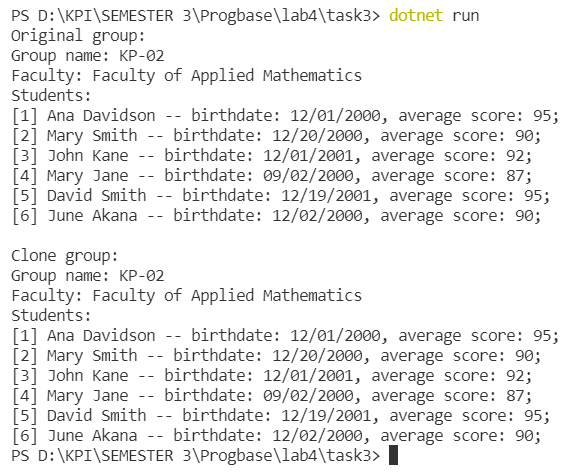
|  |
| --- |
| **GroupPrototype.cs** |
| namespace task3  {      public interface GroupPrototype      {          public StudentsGroup Clone();      }  } |

Завдання 2

|  |
| --- |
| **Program.cs** |
| class Program  {      static void Main(string[] args)      {          Director director = new Director();          Car car = director.ConstructCar("1.34BHP", "A25A", "Toyota Star");          System.Console.WriteLine(car);      }  }  class Director  {      public Car ConstructCar(string enginePower, string carcaseType, string salonType)      {          Builder builder = new CarBuilder();          builder.SetEnginePower(enginePower);          builder.SetCarcase(carcaseType);          builder.SetSalon(salonType);          return builder.GetCar();      }  }  class CarBuilder : Builder  {      private Car car;      public CarBuilder()      {          this.car = new Car();      }      public void SetEnginePower(string enginePower)      {          car.enginePower = enginePower;      }      public void SetCarcase(string carcaseType)      {          car.carcase = carcaseType;      }      public void SetSalon(string salonType)      {          car.salon = salonType;      }      public Car GetCar()      {          return this.car;      }  }  class Car  {      public string enginePower;      public string carcase;      public string salon;      public override string ToString()      {          return $"Car -- engine power: {this.enginePower}, carcase: {this.carcase}, salon: {this.salon}";      }  }  interface Builder  {      void SetEnginePower(string enginePower);      void SetCarcase(string carcaseType);      void SetSalon(string salonType);      Car GetCar();  } |

**Приклади результатів**

Завдання 1



Завдання 2



**Висновки**

Під час лабораторної роботи я ознайомилася зі породжувальними шаблонами, дізналась про особливості їх використання. Навчилась відрізняти, де саме який шаблон потрібно використовувати. А також попрактикувалась у розробленні програмного забезпечення на мові C# з використанням шаблонів Прототип та Будівельник.