ДИПЛОМЕН ПРОЕКТ

**професия код 481030 „Приложен програмист“   
специалност код 4810301 „Приложно програмиране“**

**Кристина Владиславова Велинова , 12б , N15**

**......................................................................................................................**

*/Име Презиме Фамилия на ученика, клaс, N/*

**ТЕМА**

**............................................................................................**

### Изготвил: Кристина Велинова Рецензент: ………………………

/име и фамилия/ /Иван Василев/

Перник, 2022 г.

Съдържание

1. Увод

**Увод**

Живеем във времена, в които технологиите са неизменна част от нашето ежедневие. Използваме ги постоянно, разговаряме лесно с хора, които са далеч от нас, работим от тях, търсим полезна информация в сайтове като „Google” , които улесняват много ежедневието ни. Зад всички уеб сайтове, които използваме, стоят програмисти. Всички сме наясно, че компютърното програмиране набира огромна популярност през последните години и дори десетилетия. Какво е важно да знаем за него?

Програмирането е компютърна наука и дял от математиката-информатиката, която се занимава с методите и средствата за създаване на компютърни програми.

Описано по начин, разбираем от всички може да се определи като „казване на компютъра какво да прави“ или по-точно програмистът предоставя набор от инструкции, които са написани на език за програмиране. Както при хората , така и при програмирането има различни езици. Някои от най-популярните езици за програмиране са C, C++ , C# , SQL , Java , JavaScript, Python и много други. Целта на компютърното програмиране е да създаде продукт, който може да бъде уеб страница, уебсайт, софтуерна програма, компютърна игра или дори изображение.

Защо е толкова важно да има хора, които да създават уеб сайтове?

Технологиите са толкова напреднали и вече можем да правим почти всичко онлайн. Пандемията с „Ковид-19“ засили много използването на интернет и вече можем да правим почти всичко през интернет и телефоните си. Хората започнаха да работят от вкъщи през компютрите си, учениците и студентите започнаха да се обучават онлайн. Наскоро създадоха приложения, които да улеснят поръчването на храна до вкъщи, вече започнаха и доставки от големи вериги като „Кауфланд“ , „Лидл“ , „ДМ“ и много други магазини. С днешната пандемия, работата стана много по-трудна и драстично намаля , за това онлайн сайтовете помагат на малките бизнеси да не умрат. Всичко е направено с цел улеснение за хората, както и една част от всичко за „предпазване“ от вируса.

Целта на моята дипломна работа е да разработя Уеб приложение , чиято главна цел е улеснение на потребителите, удобство и спестяване на време. Чрез нашето приложение, клиентите ни ще имат възможността да поръчат цветя, сувенири и украси , през телефоните си, без да се налага да търсят магазин. Също така в много градове в България все още нямаме обекти на място и понеже много често получаваме запитвания в социалните ни мрежи, решихме да пуснем доставки из цяла България. Това ще улесни потребителите ни да не се ангажират и да не отнемат от ценното си време в търсене на цветя или подарък за половинките си или пък за някой друг близък човек, а да могат да направят всичко много по-лесно и бързо през нашият уеб сайт. Вашата поръчка е на няколко клика разстояние, а за останалото ще се погрижим ние.

Основните цели и резултати, към които се стремим са:

1. **Проучване**

Увод в ООП (Обектnо-ориентирано програмиране)

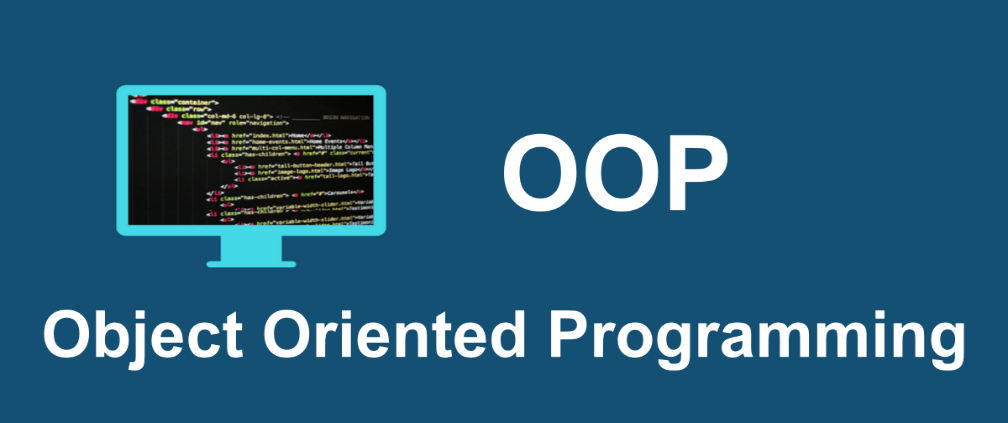
**Класове и обекти**

* Класовете са описание на реални предмети или явления наречени същности (entities). Например студент.
* Класовете имат характеристики. В програмирането са наречени свойства (properties). Например съвкупност от оценки.
* Имат поведение. В програмирането са наречени методи (methods). Например явяване на изпит.

Методите и свойствата могат да бъдат видими само в областта на класа (private-protected) или да бъдат видими публично (public).

* Обект (object) наричаме екземпляр, създаден по описанието на даден клас.

**Основни предимства и цели на ООП** – да позволи по-бърза разра­ботка на сложен софтуер и по-лесната му поддръжка. ООП позволява по лесен начин да се преизползва кода, като залага на прости и общоприети правила (принципи).



4-те основни принципа в Обектно-ориентираното програмиране са:

1. Наследяване - Inheritance  
2. Капсулация – Encapsulation

3. Полиморфизъм – Polymorphism

4. Абстракция – Abstraction



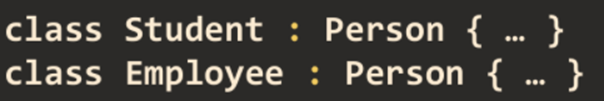
1. **Наследяване**

Наследяването е основен принцип от обектно-ориентираното програми­ране. То позволява на един клас да "наследява" (поведение и характе­ристики) от друг, по-общ клас.

Класът, който наследяваме, се нарича клас - родител или още

базов клас (base class, super class).

В C# наследяването се отбелязва чрез : оператора



В C# не се поддържа множествено наследяване. Поддържа се само имплементиране на множество интерфейси.

Модификатори за достъп на членове на класа при наследяване

Когато се наследява един базов клас:

* Всички негови public , protected и protected internal членове

(методи, свойства и т.н.) са видими за класа наследник.

* Всички негови private методи, свойства и член-променливи не са

видими за класа наследник.

* Всички негови internal членове са видими за класа наследник само

ако базовият клас и наследникът са в едно и също асембли.

1. **Абстракция**

Абстракцията означава да се игнорират несъществените черти, свойства или функции и да се наблегне на съществените за проекта, което помага да се управлява сложността.

Тя ни позволява да пишем код, който работи с абстрактни структури от данни (например списък, речник, множество и други). Имайки абстрактния тип данни, ние можем да работим с него през неговия интерфейс, без да се интересуваме от имплементацията му.

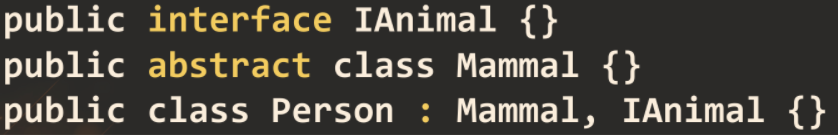
Абстракцията ни позволява и нещо много важно – да дефинираме интерфейс на нашите програми, т.е. да дефинираме всички задачи, които тази програма може да извърши, както и съответните входни и изходни данни.

Дефинира се с ключовата дума abstract.

Ако един клас е абстрактен, то той не може да бъде инстанциран, т.е. не може да се създават обекти от този клас.

Всеки клас, който има поне един абстрактен метод, трябва да бъде абстрактен.

Как постигаме абстракция?



1. **Полиморфизъм**

Полиморфизмът в [обектно ориентираното програмиране](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) представлява свойството на [обектите](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) от един и същи тип да имат един и същи [интерфейс](https://bg.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1), но с различна реализация на този интерфейс.

Различни неща или обекти могат да имат еднакъв интерфейс или да отговарят на едно и също (по наименование) съобщение и да реагират подходящо, в зависимост от природата или типа на обекта. Това позволява много различни неща да бъдат взаимозаменими.

Например, ако една птица получи съобщение „движи се бързо“, тя ще маха с крила и ще лети. Ако един лъв получи същото съобщение, той ще тича, използвайки краката си. И двете животни отговарят на една и съща молба по начини, които са подходящи за всяко от тях.

Смисълът на полиморфизма може да бъде накратко изразен със следната фраза – „Един интерфейс, множество от различни реализации“. Чрез полиморфизма се постига по-голяма абстракция и по-лесно повторно използване на кода.

1. **Капсулация**

Означава скриване на ненужните детайли за обектите и откриване към външния свят само на важните техни характеристики и свойства.

Идеята за капсулацията е да се затворят (тоест, капсулират) свойствата и методите на абстракцията, и да се излагат (expose) само тези, които са абсолютно необходими.

Капсулацията обслужва три полезни цели:

· Елиминира външно влияние върху кода и злоупотребата с данни, вградени в него.

· Допуска включване на валидизиращ код, за да се хващат грешките при използване на публичния интерфейс. Примерно проверка да не се въведе отрицателно число за заплата на служител.

· Освобождава потребителя от задължението да знае как са вложени свойствата и методите.

Капсулиране - начин да се опрости процеса на кодиране.