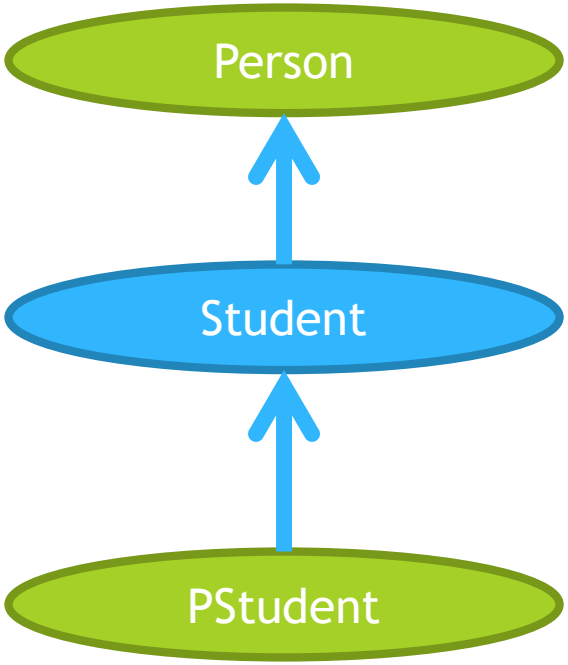
Статично и динамично свързване. Полиморфизъм. Абстрактен клас. Виртуални деструктори.

доц. д-р Нора Ангелова

Статично и динамично свързване

Статично и динамично свързване

- Наследяване



Статично и динамично свързване

- Статично свързване - изборът на функцията, която трябва да се изпълни става по време на компилация.

```
Student st("Student1", "890504", "44394", "str...");
PStudent pst("Student1", "890504", "44394", "str...",
6.0, "Astea Solutions");

Person* pstPtr = &pst; // Възможно ли е? Защо?
pstPtr->print();       // Коя функция ще се извика?

Person* stPtr = &st;
stPtr->print();
```

Статично и динамично свързване

- Статично свързване - изборът на функцията, която трябва да се изпълни става по време на компилация.

```
Student st("Student1", "890504", "44394", "str...");
PStudent pst("Student1", "890504", "44394", "str...",
6.0, "Astea Solutions");

Person* pstPtr = &pst;
pstPtr->print();        // Person::print

Person* stPtr = &st;
stPtr->print();         // Person::print
```

Статично и динамично свързване

- Динамично свързване – изборът на функцията, която трябва да се изпълни става по време на изпълнение на програмата.

Статично и динамично свързване

Динамично свързване

- Разширяването на йерархията не създава проблеми.
- Не се налага проверка на типа.
- Усложняване на кода и забавя процеса на изпълнение на програмата.
- Реализира се чрез специални член-функции на класове – виртуални член-функции.
- Виртуалните функции се декларират чрез поставяне на запазената дума virtual.

virtual <тип_на_резултата> <име_на_метод>(<параметри>);

Статично и динамично свързване

Динамично свързване

```
// Декларация на print в Point2D, Point3D, ColPoint3D
virtual void print() const;

Student st("Student1", "890504", "44394", "str...");
PStudent pst("Student1", "890504", "44394", "str...",
6.0, "Astea Solutions");

Person* pstPtr = &pst;
pstPtr->print();        // PStudent::print
```

- Ще се определи по време на изпълнението на програмата
- Определянето е в зависимост от класа на ОБЕКТА

```
Person* stPtr = &st;
stPtr->print();         // Student::print
```

Статично и динамично свързване

Динамично свързване

1. Само член-функциите на класовете могат да се декларират като виртуални.
2. Ако функция е обявена за виртуална в основния клас, декларираните член-функции в производните класове със същия прототип също са виртуални дори ако запазената дума бъде пропусната.
3. Ако в производен клас се дефинира виртуална функция, която има същия прототип като невиртуална функция в основния клас, двете функции се интерпретират като различни член-функции.
4. Възможно е виртуална функция да се дефинира извън клас. Тогава не започва със запазената дума virtual.
5. Виртуалните член-функции се наследяват като останалите компоненти на класа.
6. Достъпът до съответните функции се определя спрямо типа на указателя, а не спрямо типа на обекта към който е насочен.
7. Основният клас, в който член-функция е обявена за виртуална, трябва да е с атрибут public в производните от него класове.
8. Виртуалните член-функции се извикват чрез указател или псевдоним на обект на някакъв клас.
9. Виртуалната член-функция, която в действителност се изпълнява, зависи от класа на обекта, към който сочи указателят.

Статично и динамично свързване

Статично vs Динамично свързване

- С виртуални функции е възможно в базов клас да се извика метод на производен клас
- Съществуват три случая, при които обръщение към виртуална член-функция се разрешава статично:
  - Виртуалната функция се извиква чрез обект на класа, в който е дефинирана.
    ```
    Student st1;
    st1.print();
    ```
  - Виртуалната член-функция се активира чрез указател към или чрез псевдоним на обект, но явно, чрез оператора ::, е посочена конкретната виртуална член-функция.
    ```
    Person *ptr= &st1;
    ptr->print();              // динамично свързване
    ptr->Person::print();      // статично свързване
    ```
  - Виртуалната член-функция се активира в тялото на конструктор или деструктор на основен клас.

    В този случай се изпълнява виртуалната член-функция на основния клас. Това е така, защото виртуалната функция в конструктора или деструктора на основния клас се извиква когато обектът от производния клас още не е създаден или вече е разрушен.

ПОЛИМОРФИЗЪМ

- Едни и същи действия се реализират по различен начин в зависимост от обектите, върху които се прилагат.
- Действията се наричат полиморфни.
- Свойство на член-функциите на класовете.
- Реализира се чрез виртуални функции.
- Класовете, върху които ще се прилага, трябва да имат общ родител или прародител, т.е. да са производни на един и същ клас.
- В класа се дефинира виртуален метод, съответстващ на полиморфното действие.
- Всеки клас предефинира или не виртуалния метод.
- Активирането става чрез указател към базов клас, на който може да се присвоят адресите на обекти на който и да е от производните класове от йерархията.
- Ще се изпълни методът на съответния обект.

Абстрактен клас

- Ако класовете, в които трябва се дефинират виртуални методи, нямат общ родител, такъв може да бъде създаден изкуствено чрез т.нар. абстрактен клас.

- Клас, в който има поне една чисто виртуална функция.

```
virtual <тип_на_резултата> <име_на_метод>(<параметри>) = 0;
```

- Не могат да се създават обекти от тези класове, но могат да се дефинират указатели към такива класове.
- Чисто виртуалните функции задължително трябва да бъдат предефинирани в производните класове или да бъдат обявени като чисто виртуални в тях.

Виртуални деструктори

```
Person* personPtr = new Student();

delete personPtr; // Какво се разрушава?
```

ВИРТУАЛНИ ДЕСТРУКТОРИ

```
class Person {
public:
  Person(const char* = "", const char* = "");
  virtual ~Person();
  Person(Person const&);
  Person& operator=(Person const& p);

  // Член-функция за извеждане
  virtual void print() const;
private:
  char* name; // име
  char* ucn; // ЕГН
  // Помощни член-функции за копиране и изтриване
  void copyPerson(const char*, const char*);
  void delPerson();
};
```

* Деструкторите на всички класове в йерархията ще бъдат виртуални

Какво може да правим с тези знания

- Обединяване на класове обща йерархия
- Създаване на хетерогенни контейнери

ВРЕМЕ
ЗА ВАШИТЕ ВЪПРОСИ