

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Основы ООП

ВОПРОСЫ

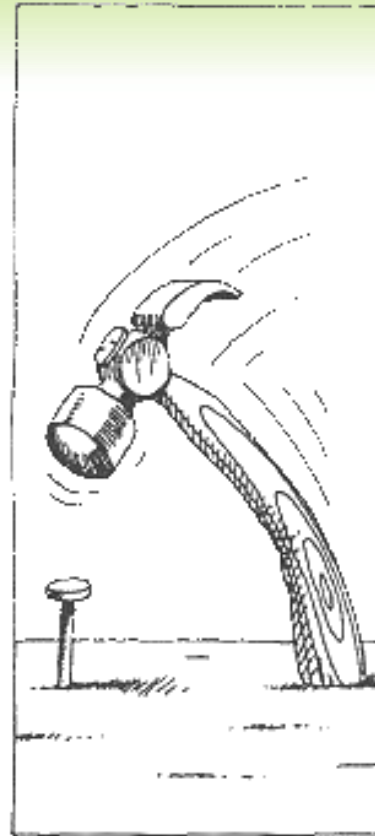
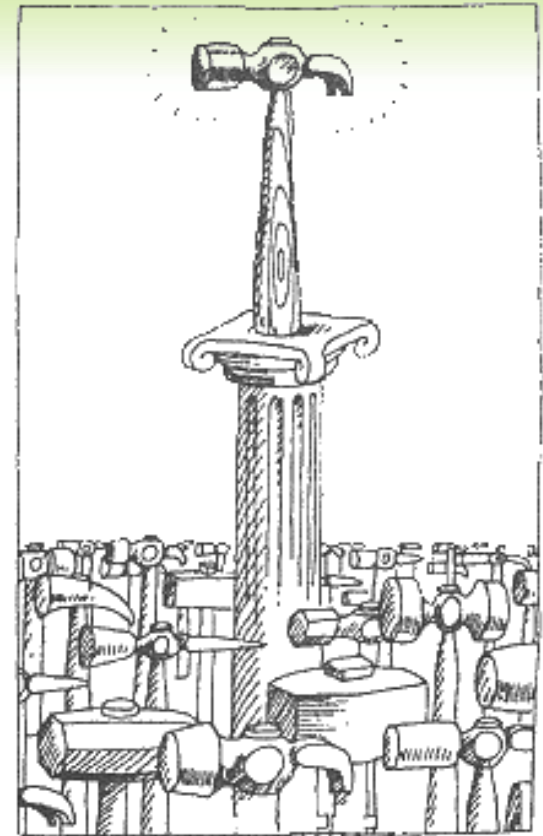
- ① Какие проблемы возникают при попытке разработки программного обеспечения?
- ① Какие есть методы решения проблем, возникающих при разработке программного обеспечения?
- ① Какие методы декомпозиции применяются в современных языках программирования?

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Объектно-ориентированное программирование - это методология программирования, основанная на представлении программы в *виде совокупности объектов*, каждый из которых является *экземпляром определенного класса*, а классы образуют *иерархию наследования*.

- ◎ Объект - это нечто, имеющее четко определенные границы...
- ◎ но этого недостаточно, чтобы отделить один объект от другого или дать оценку качества абстракции.
- ◎ **Объект** обладает состоянием, поведением и идентичностью; структура и поведение схожих объектов определяет общий для них класс; термины "экземпляр класса" и "объект" взаимозаменяемы.

- ◎ Класс – это множество объектов, обладающих общей структурой, поведением и семантикой.
- ◎ Отдельный объект – это экземпляр класса.
- ◎ Класс представляет лишь абстракцию существенных свойств объекта.

**Состояние****Поведение****Индивидуальность**

СОСТОЯНИЕ ОБЪЕКТА

- ◎ Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.

Например:

- ◎ торговый автомат имеет свойство: способность принимать монеты
- ◎ этому свойству соответствует динамическое значение – количество принятых монет

ПРИМЕР ОПИСАНИЯ СОСТОЯНИЯ НА C++

```
struct PersonnelRecord {  
    char name[100];  
    int socialSecurityNumber;  
    char department[10];  
    float salary;  
};
```

ПОВЕДЕНИЕ ОБЪЕКТА

- ◎ **Поведение** - это то, как объект действует и реагирует; поведение выражается в терминах состояния объекта и передачи сообщений.
- ◎ Операцией называется определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию.

Например:

- ◎ клиент может активизировать операции `append()` и `pop()` для того, чтобы управлять объектом-очередью.

ПРИМЕР ОПИСАНИЯ ПОВЕДЕНИЯ НА C++

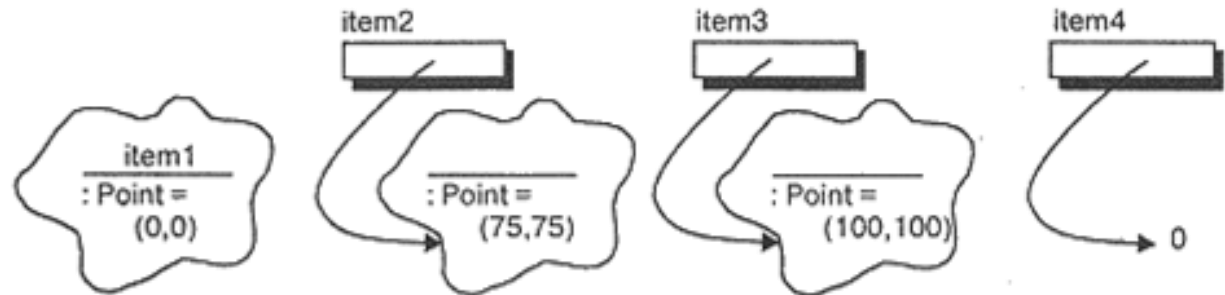
```
class Queue {  
public:  
    Queue();  
    Queue(const Queue&);  
    virtual ~Queue();  
    virtual Queue& operator=(const Queue&);  
    virtual int operator==(const Queue&);  
    int operator!=(const Queue&) const;  
    virtual void clear();  
    virtual void append(const void*);  
    virtual void remove(int at);  
    virtual int length() const;  
    virtual int isEmpty() const;  
    ...  
};
```

ИНДИВИДУАЛЬНОСТЬ ОБЪЕКТА

- ◎ Индивидуальность - это такое свойство объекта, которое отличает его от всех других объектов
- ◎ В большинстве языков программирования при создании объект именуется, поэтому многие путают *адресуемость* и *индивидуальность*
- ◎ Невозможность отличить имя объекта от самого объекта является источником множества ошибок в ООП

ПРОБЛЕМА ИДЕНТИЧНОСТИ (1)

```
DisplayItem item1;  
DisplayItem* item2 = new  
                    DisplayItem(Point(75, 75));  
DisplayItem* item3 = new  
                    DisplayItem(Point(100, 100));  
DisplayItem* item4 = 0;
```



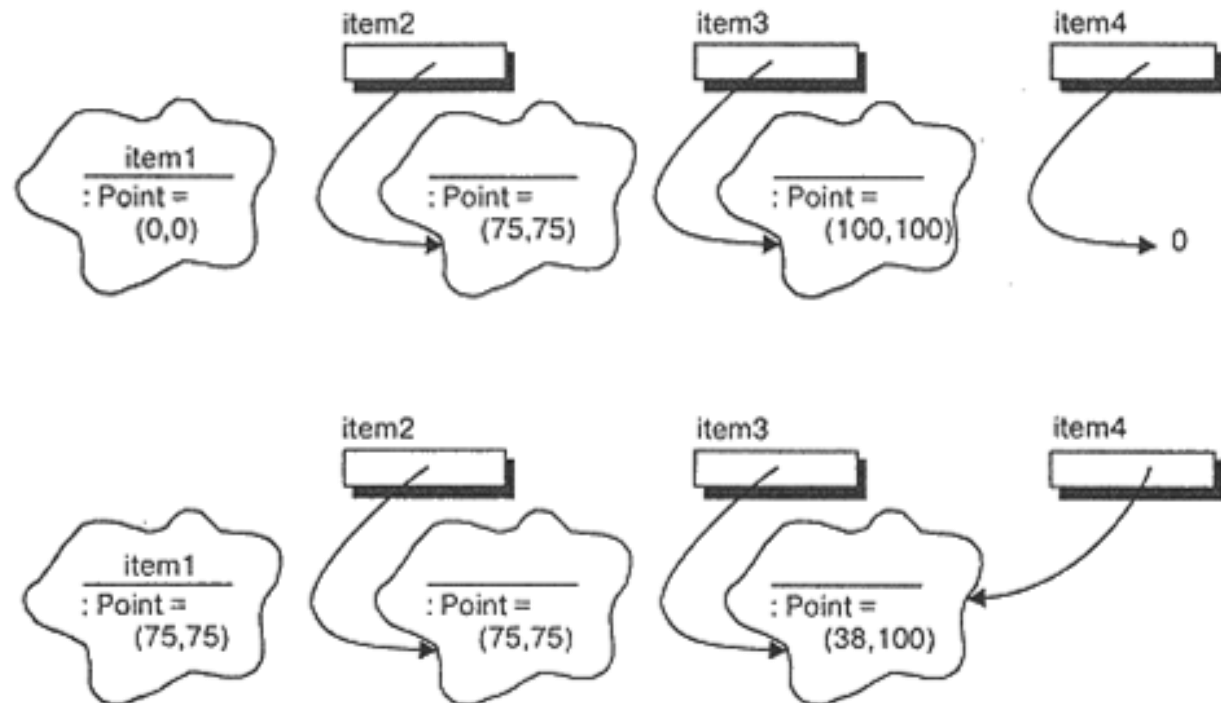
ПРОБЛЕМА ИДЕНТИЧНОСТИ (2)

14

```
item1.move(item2->location());
```

```
item4 = item3;
```

```
item4->move(Point(38, 100));
```

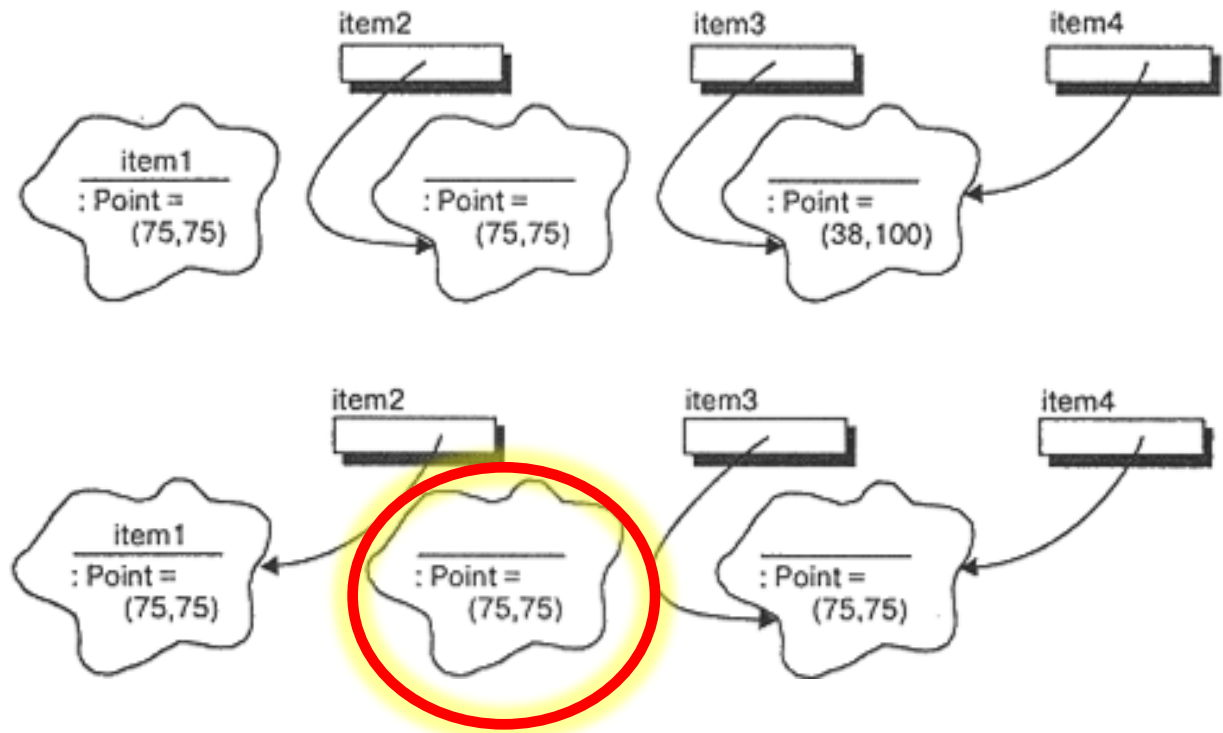


ПРОБЛЕМА ИДЕНТИЧНОСТИ (3)

15

```
item2 = &item1;
```

```
item4->move(item2->location());
```



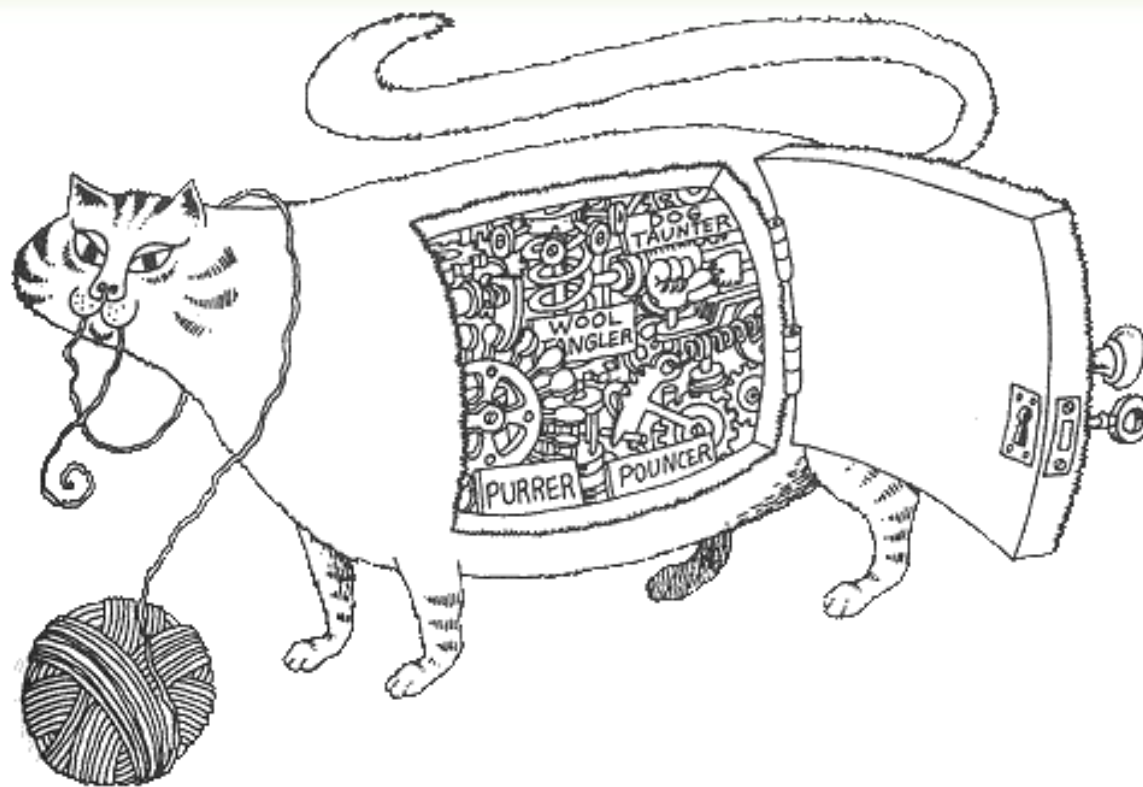
3 КИТА ООП

1. Инкапсуляция
2. Наследование
3. Полиморфизм

ИНКАПСУЛЯЦИЯ

- ◎ **Инкапсуляция** - это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.
- ◎ Это механизм, который объединяет данные и код, манипулирующий этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования.

ИНКАПСУЛЯЦИЯ

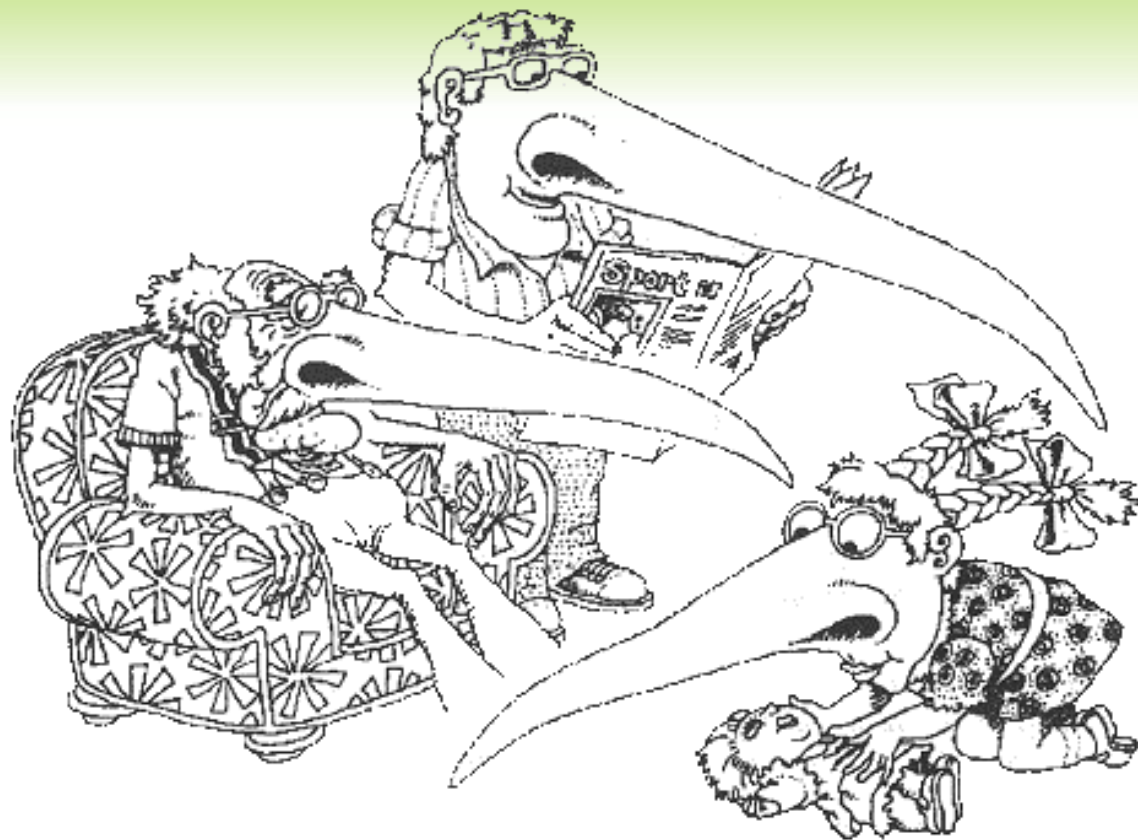


Инкапсуляция скрывает детали реализации объекта.

НАСЛЕДОВАНИЕ

- ⊙ Наследование - это процесс, посредством которого один объект может приобретать свойства другого. Точнее, объект может наследовать основные свойства другого объекта и добавлять к ним черты, характерные только для него.
- ⊙ Наследование является важным, поскольку оно позволяет поддерживать концепцию иерархии классов. Применение иерархии классов делает управляемыми большие потоки информации.

НАСЛЕДОВАНИЕ



Дочерний класс может унаследовать структуру и поведение родительских классов.

ПОЛИМОРФИЗМ

- ⊙ **Полиморфизм** - это свойство, которое позволяет использовать *одно и то же имя* для решения *нескольких* схожих, но технически разных задач.
- ⊙ В более общем смысле, концепцией полиморфизма является идея «один интерфейс, множество методов». Полиморфизм помогает снизить сложность программ, разрешая использование того же интерфейса для задания единого класса действий в иерархии классов. Выбор же конкретного действия, в зависимости от ситуации, возлагается на компилятор.
- ⊙ В C++ можно использовать одно имя функции для множества различных действий. Это называется перегрузкой функций.

ПОЛИМОРФИЗМ



Полиморфизм позволяет по-разному реализовать одну и ту же операцию в дочерних объектах

- ◎ ООП – это методология программирования, основанная на представлении программы в виде совокупности объектов.
- ◎ Объект – имеет состояние (поля), поведение (методы), индивидуальность (имя).
- ◎ 3 кита ООП: инкапсуляция, наследование полиморфизм:
 - ◎ Инкапсуляция – ограничение доступа к внутренней реализации объекта
 - ◎ Наследование – способность производить новый класс из существующего базового класса.
 - ◎ Полиморфизм – позволяет использовать *одно и то же имя* для решения *нескольких* схожих, но технически разных задач.