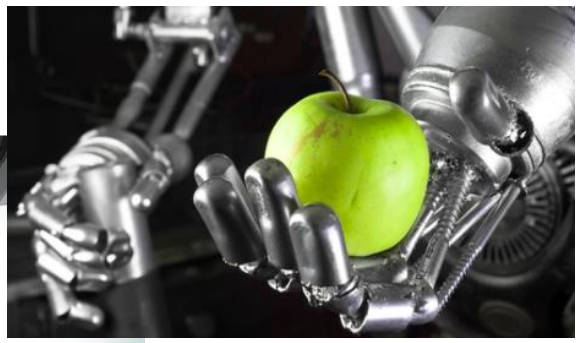
```
/*
 * advanced programming
 * Alireza Akhavan Pour
 * akhavan@alirezaWeb.com
 * date: 1395/01/30
 */
int main()
{
      cout<< "In the name of God";
      Lecture_18();
      return 0;
}

string Lecture_18()
{
      cout << "1.review: UML & Inheritance" <<endl;
      cout << "2.dynamic binding VS static binding" <<endl;
      cout << "3.Introduction to polymorphism" <<endl;
      return ":)"
}
```

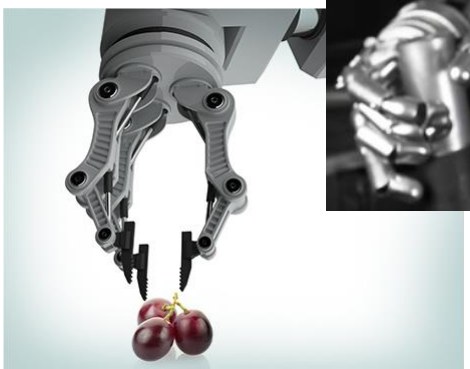
Shahid Rajaee Teacher Training University

1

polymorphism

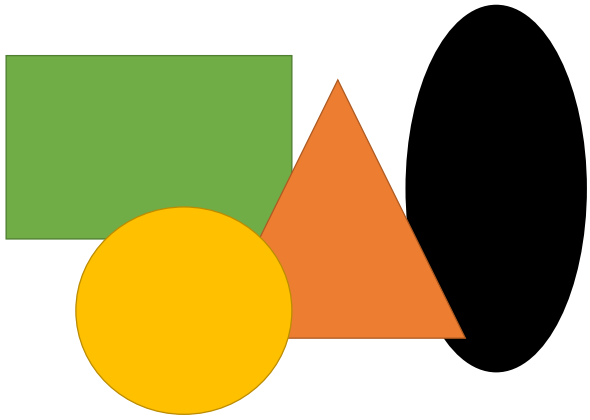
چندریختی





2

مثال رسم شکل‌های هندسی



3

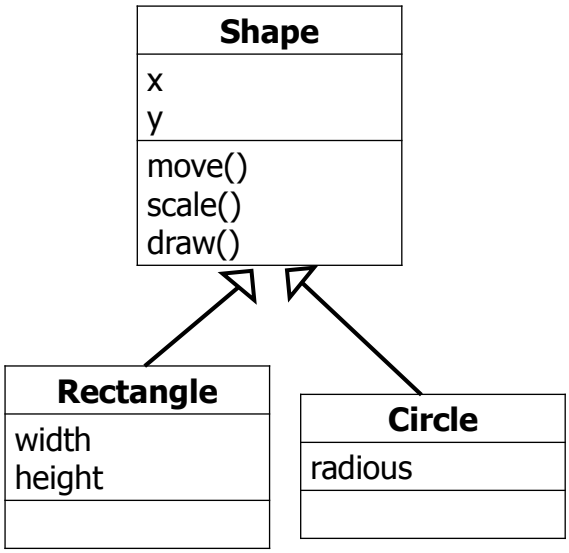
کلاس‌های متناظر با شکل‌ها

Rectangle
x y width height
move() scale() draw()

Circle
x y radious
move() scale() draw()

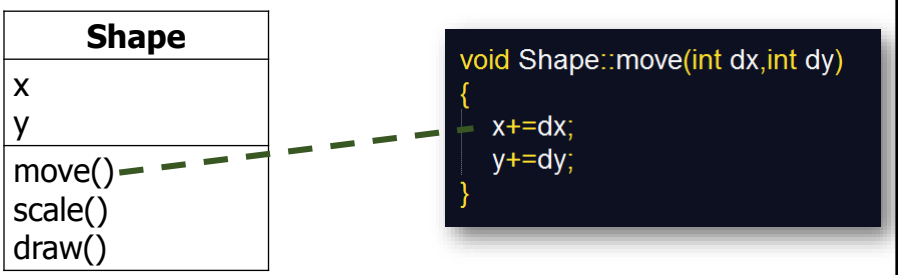
4

فاکتورگیری اعضای مشترک



5

پیاده سازی move



6

پیاده سازی scale

Shape
x y
move() scale() draw()

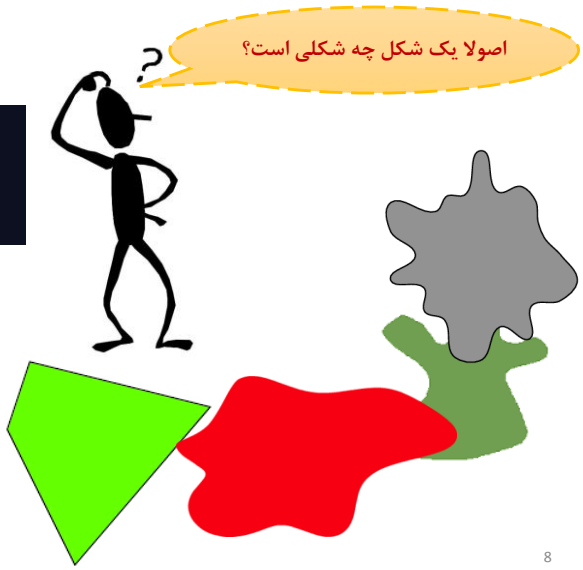
پیاده سازی این متد در کلاس shape قابل انجام نیست
و باید توسط زیر کلاس‌ها صورت پذیرد.

7

استفاده از Shape

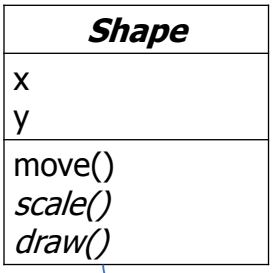
یک شکل!

```
Shape s(10,30);
s.move(2,-3);
s.scale(5); //!
```



8

کلاس‌های مجرد یا انتزاعی (abstract)



کلاس مجرد:
کلاسی که حداقل یک متد مجازی خالص داشته باشد.

```
class Shape
{
public:
    Shape(int, int);
    void move(int dx, int dy);
    virtual void scale(double s) = 0;
    virtual void draw() = 0;
};
```

کلاس مجازی خالص (pure virtual):
متدی که بدنه ندارد و پیاده سازی آن به زیر کلاس‌ها واگذار شده است.

9

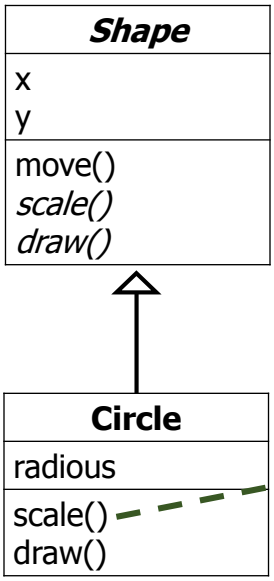
ساختن شئ از کلاس مجرد ممکن نیست!

```
Shape s(10, 30);
```

Compile error!!

10

مسئولیت زیر کلاس‌ها

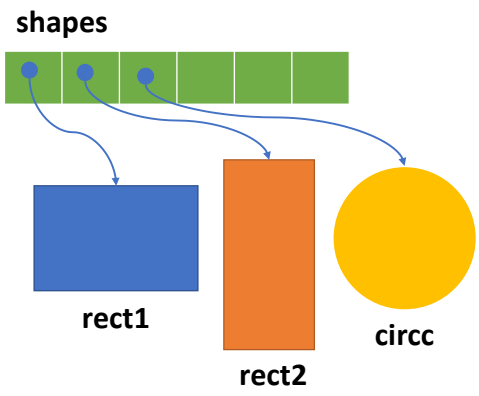


زیرکلاس‌ها باید متدهای مجرد را پیاده سازی کنند
وگرنه خود نیز مجرد محسوب می‌شوند.

```
void Circle::scale(double factor)
{
    radious *= factor;
}
```

11

نگهداری شکل ها

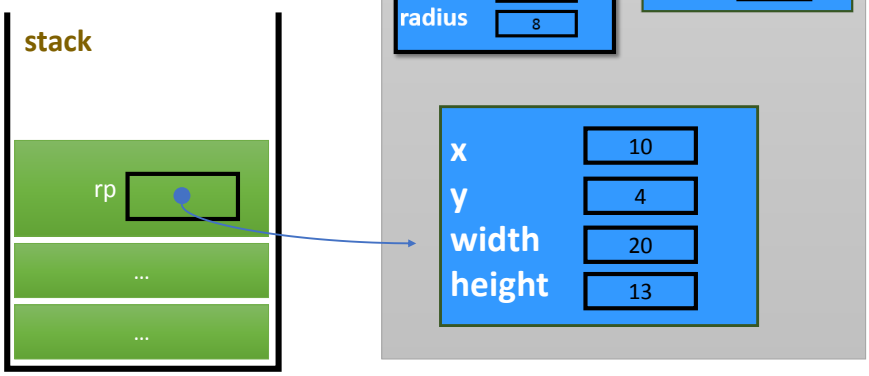



```
vector <Shape*> shapes;

Rect rect1(10,7,3,2);
Rect rect2(3,15,1,4);
Circle circ(3,2);

shape.push_back(&rect1);
shape.push_back(&rect2);
shape.push_back(&circ);
```

12

تخصیص حافظه‌ی پویا

```
Rect* rp= new Rect(...);
...
delete rp;
```



13

آزاد کردن حافظه ی تخصیص یافته

- چرا باید حافظه ای را که به صورت پویا میگیریم حتما آزاد کنیم؟!
  - ❑ استفاده بهینه از حافظه در طول اجرا
  - ❑ اتکا نکردن به سیستم عامل برای آزاد کردن حافظه پس از اتمام اجرا

14

انضباط در مدیریت حافظه

- حافظه ای را که به طور پویا گرفتید حتما آزاد کنید
- حافظه ایرا که با new گرفته اید با delete و حافظه ای را که با malloc گرفته اید با free آزاد کنید.
- حافظه ای را دوبار آزاد نکنید!

15

مثال چندریختی

- در این مثال، اشکال هندسی (به عنوان نمونه، مستطیل و دایره) در سلسله‌مراتب وراثت از کلاس مجرد «شکل» به ارث می‌برند.

Shapes.cpp

16

منابع این جلسه

- http://ramtung.ir/apnotes/html/11_Inheritance.html

17