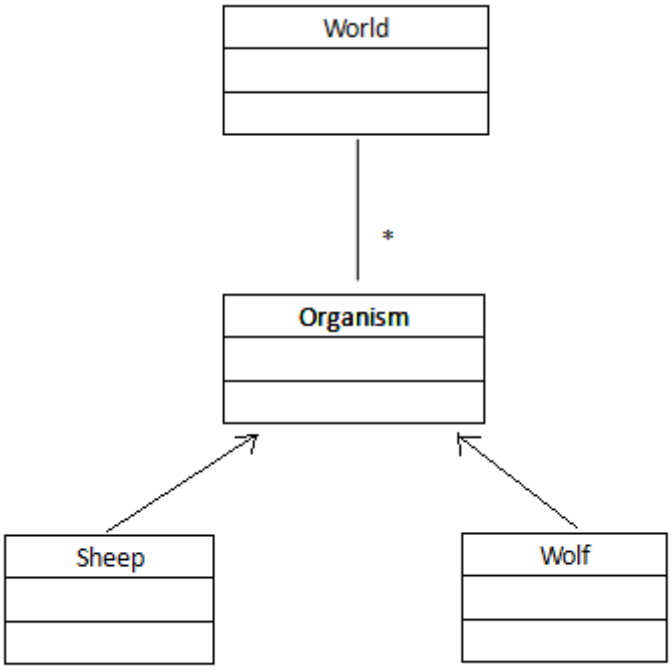
一、 类图



二、 生物体类及其派生类

生物体是抽象的概念，定义该类的目的并不是要生成生物体对象，而是为了把它作为派生其他类的基类，并通过该基类访问派生类对象。

所有的生物体都有一些公有行为，例如吃、移动、繁殖、死亡。但是这些公共行为针对不同的生物，成员函数的实现是不同的。因此抽象类 Organism 中，这些成员函数应该设置为纯虚函数。

生物体都必须放置在 2D 网格空间中，并且一个网格中只能有一个生物体，因此在生物体移动和繁殖时，都要知道其邻域中是否有生物体，该生物体是什么，因此需要有一个成员函数 getType() 获取任何一个（x,y）位置处的生物类。

构造函数的设计，除了有无参的默认构造函数外，还需要创建一个特定位置的生物体。

```
public:
    Organism();
    Organism(World *world, int x, int y);
    ~Organism();
    virtual void breed() = 0;   // Whether or not to breed
    virtual void move() = 0;    // Rules to move the critter
    virtual int getType() =0;   // Return if ant or doodlebug
    virtual bool starve() = 0;  // Determine if organism starves
```

生物体的数据成员：

World 中的位置信息（x，y），至于其他的数据成员，需要在具体实现时，再添加。

Organism {*Abstract*}
world: world* # x: int # y:int …
Organism(world*,int ,int) + breed() = 0 : void + move() = 0: void + getType() = 0: int + starve() =0 : bool

由基类 Organism 派生出 Sheep 类和 Wolf 类，在派生类中重新定义基类的纯虚函数的实现代码

三、 World 类：

这个类是展现模拟的 2D 网格空间。模拟中主要两个行为：SimulateOneStep 和 Display。

- Display ： world 中如何显示捕食者和被捕食者。在控制台程序中，采用 ASCII 码形式来表示，用“o”、“X”分别表示羊和狼

  思考：如何在图形界面下显示羊和狼？

- SimulateOneStep ：这个是模拟中的主要函数，该函数是在一个 step 中，定义捕食者和被捕食者的行为规则。

- 每个 step 中，按照 world 从左到右，从上到下的顺序，狼先移动，羊后移动。

o				
	X		o	
		o		
	o		X	
			o	

例如在 5*5 的网格单元中，按照从左到右，从上到下的顺序，狼先移动。grid[1][1]处的生物类型是狼，其上下左右都为空，随机移动到其中任一网格单元，假设移动到 grid[1][2]。如何实现这一移动过程？grid[1][1]处的生物类型为 NULL，grid[1][2]处的生物类型为 Wolf。羊的移动同理可得。

在模拟空间中需要把生物体设置在特定位置，并且需要获取特定位置处的生物体，因此需要设计两个成员函数函数

- void setAt (int x, int y, Organism* org): 把生物体指针 org 指向的生物放置在特定的位置（x,y）处；
- Organism* getAt(int x, int y) ： 获取特定位置（x,y）处的生物体的指针

World 类的数据成员：

比较简单明确，可以用一个二维数据 grid[ ][ ]表示，且这个数组的类型应该是基类指针 Organism*

World
- grid[][]: Organism*
+getAt(int ,int): Organism* +setAt(int,int ,Organism*):void +Display(): void +SimulateOneStep(): void

1. 可以用 STL 中的 vector 向量容器、list 双向链表来存放羊和狼？

2. 利用位图显示羊和狼

（1） 在资源视图中添加已经绘制好的位图背景、羊和狼，并分别命名为 IDB_BITMAP0，IDB_BITMAP1，IDB_BITMAP2;

（2） 在 View 类的.h 文件中定义变量：

```
public:
	CBitmap bitmap0,bitmap1,bitmap2;
```

（3） 在 View 类中的 LBUTTONDOWN 中，实现每次单击鼠标左键，实现一次 timestpe 操作

```
CDC* pDC = GetDC();
CDC dc;
dc.CreateCompatibleDC(pDC);
dc.SelectObject (&bitmap1);
BITMAP bm;
bitmap1.GetBitmap(&bm);
CRect rect;
GetClientRect(&rect);

int W = rect.Width() , H = rect.Height();
int w = bm.bmWidth,h = bm.bmHeight;

int i,j;
int grid[20][20];
//贴羊的位图
int countSheep =20;
i =0;
while ( i < countSheep)
{
  int x = rand()% 20;
  int y = rand() % 20;
  if (grid [x][y] == 0 )
  {
	dc.SelectObject(&bitmap1);
	pDC ->BitBlt (x*w,y*h,w,h,&dc,0,0,SRCCOPY);
	i++;
  }

}
```