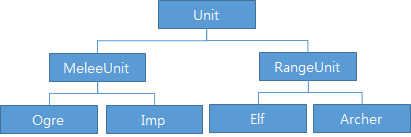
1. **2차원 평면 상의 한 점의 좌표를 표시하기 위한 좌표 클래스를 구현하라**.(50점)   
    아래 내용 중 어떤 항목을 구현 했는지에 대한 설명을 comment 로 작성하고, 구현한 기능들이 잘 수행되는지 증명할 수 있는 main 코드 작성. 여기서 기능들 수행 여부가 증명되지 않으면 점수를 못 받을 수도 있음. 이 main 코드 부분 자체도 점수에 반영 (5점)  
   좌표는 기본적으로 2차 평면 좌표계 상에서 x축, y축의 값을 가지는 2차원 백터 형태로 표현된다. (예, <2,4>, <-3.5,5>)
   1. 좌표클래스 Coord는 x, y 좌표 외에 아래 기능들을 하는 멤버함수들로 구성 (2점)
   2. Template 클래스로 만들어 x, y좌표에 float, double, int 타입이 들어갈 수 있어야 함. (8점)
   3. 아래 코드가 동작하기 위한 constructor들 (6점)  
      Coord<double> p1; // <0,0>  
      Coord<double> p2(3.14); // <3.14,0>  
      Coord<double> p3(1.5, 3.14); // <1.5,3.14>  
      Coord<double> p4(p2); // <3.14,0> (p2와 동일)
   4. x혹은 y좌표를 리턴해 주는 [ ] 오퍼레이터와, 아래의 예와 같이 x, y가 index로 들어갈 수 있게 해 주는 enum 타입 정의 (enum 타입을 사용하지 않고 숫자를 사용해도 부분점수 있음) (5점)  
      p1[x] = p2[y] //p2의 y좌표를 p1의 x 좌표에 어싸인  
      p1[x] = p2[x]+p3[x] //p2의 x좌표와 p3의 x좌표를 더해 p1의 x 좌표에 어싸인
   5. 다음 코드가 동작하기 위한 assignment operator (2점)  
      p2 = 1.4 //p2에 <1.4,0> 이 저장
   6. 두 좌표를 더하거나 뺄 수 있게 해 주는 외부 operator overloading 함수 (+ 와 -). 좌표간의 더하기와 빼기는 x, y 값을 각각 더하거나 빼 주면 됨. (4점)  
      예) <2,5> - <2,4> 🡺 <0,1>
   7. Coord 와 실수간의 곱하기와 나누기 연산자 overloading (\* 와 /). x, y 좌표 모두에 해당 실수를 곱하거나 나누면 됨. (3점)  
      예) p1 = p2\*4 //p2가 <1,2> 였다면 p1에는 <4,8> 이 저장  
       p1 = p2/2 //p2가 <2,4> 였다면 p1에는 <1,2>가 저장
   8. Coord와 Coord 간의 곱하기연산자 overloading (\* 연산자). 이것은 2차원 백터간의 내적으로 계산함. p1과 p2의 내적은 <p1.x\*p2.x , p1.y\*p2.y> 임. (3점)
   9. 좌표와 원점<0,0> 까지의 거리 계산하는 함수 norm 구현. 거리 계산은 sqrt(x2 + y2)임. (3점)
   10. 나눗셈 시 나누기 0 exception을 throw-catch 로 구현 (4점)
   11. 좌표를 <x, y> 형식으로 출력해주는 함수. (1점)
   12. 좌표 비교를 위한 ==, != 비교 연산자 오버로딩 (x, y값 둘 다 같아야 함) (4점)
2. **스타 크래프트와 같은 전략게임의 유닛을 정의하는 클래스들을 디자인 하라. (50점)**
   1. 다음 요구사항들을 만족하게 클래스를 디자인 하라. 요구사항 외의 사항들은 자유롭게 디자인 하라.
      1. 다음과 같은 구조로 클래스 간의 상속관계를 정의하라.



* + 1. Unit 클래스: Abstract 클래스
       1. 데이터  
          유닛의 2차원 좌표: 1번에서 정의한 Coord를 사용하라 (아니면 감점)  
          현재 에너지, 공격력, 방어력, 이동력 (우선 모두 0으로 초기화)  
          현재까지 생성된 Unit 객체의 개수를 카운팅 하는 Static 변수 하나 생성하고 생성자에서 갱신해 줌
       2. 함수  
          Move: Coord 를 입력받아 그 방향으로 이동력 만큼 좌표 이동하는 함수  
          예) <1,1>이 현재 좌표이고 목표 좌표가 <4,1>이고 이동력이 2이면 함수 호출 후 좌표는 <3,1>이 되어 있어야 함.   
          만약 이동력이 목표좌표와 현재좌표간의 거리보다 커서 목표좌표를 지나치게 되면 목표좌표까지만 이동시킴. 예) <1,1>이 현재 좌표이고 목표 좌표가 <4,1>이고 이동력이 6이면 함수 호출 후 좌표는 <4,1>이 되어 있어야 함.  
          Attack: 각 유닛마다 공격 방법이 다르므로 순수가상함수로 정의
    2. MeleeUnit: 붙어서 싸우는 유닛.
       1. 상속받은 Attack 함수를 다음과 같이 정의
          1. 공격받는 Unit 을 입력 파라미터로 받음.
          2. 공격 시 룰: 입력 Unit의 위치와 자신과의 거리 (Coord의 – operator와 norm 함수로 거리 계산 가능)가 5 이내일 때만 공격가능. 아니면 “공격불가” 메시지 출력하고 공격하지 않음. 공격받는 Unit의 방어력이 공격하는 Unit의 공격력보다 작으면 공격력만큼 공격받는 Unit의 에너지를 감소시키고, 방어력이 더 크면 공격력/2만큼 공격받는 Unit의 에너지 감소시킴. 에너지가 0이하가 되면 메시지 출력하고 유닛 사라짐
       2. SpecialMeleeAttack: 순수 가상 함수
    3. RangeUnit: 멀리서 공격하는 유닛
       1. 얼마나 멀리 공격할 수 있는지 저장하는 변수 “range” 정의 (0으로 초기화)
       2. 상속받은 Attack 함수를 다음과 같이 정의
          1. 공격받는 Unit 을 입력 파라미터로 받음.
          2. 공격 시 룰: 입력 Unit의 위치와 자신과의 거리 (Coord의 – operator와 norm 함수로 거리 계산 가능)가 “range” 이내일 때만 공격가능. 아니면 “공격불가” 메시지 출력하고 공격하지 않음. 공격받는 Unit의 공격력만큼 공격받는 Unit의 에너지를 감소시킴. 에너지가 0이하가 되면 메시지 출력하고 유닛 사라짐
       3. SpecialRangeAttach: 순수 가상함수
    4. Ogre, Imp, Elf, Archer 유닛
       1. 각각의 에너지, 공격력, 방어력, 이동력 및 RangeUnit의 range 값을 생성자에서 초기화 (각 숫자는 알아서 작성)
       2. 위치를 입력 받아 이로 유닛의 위치를 초기화 하는 생성자 구현
       3. Ogre와 Imp의 SpecialMeleeAttack을 정의
          1. Ogre에서는 일시적으로 공격력을 2배로 하여 attack
          2. Imp는 2번 공격
       4. Elf와 Archer의 SpecialRangeAttach을 정의
          1. Elf는 range를 2배로 하여 공격
          2. Archer는 2번 연속 공격
    5. Main 에서는 다음을 보여줄 수 있는 코드 작성
       1. Unit을 가리키는 포인터를 사용하여 Dynamic binding 을 하는 예제코드 작성 (10점)
       2. 모든 4종류의 Unit들의 공격, 이동, 공격불가, SpecialAttack 등이 잘 동작하는지 검증하는 코드 (30점)
       3. 한 유닛을 Binary 파일 입출력으로 파일에 저장하고 다시 불러오는 과정을 수행하고 검증하는 코드 작성 (10점)