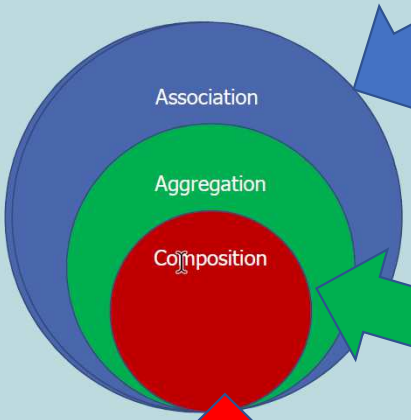
Kế Thừa

Phần 2

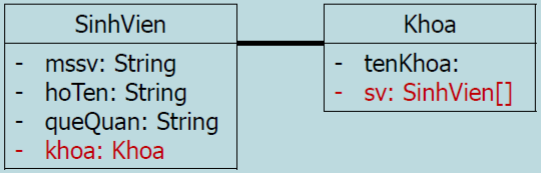
- Một số ký hiệu UML bổ sung (sv cần xem them trong tài liệu học tập)
- Nhắc lại 1 số điểm về kế thừa.
- Thử nghiệm việc gọi ngầm định không tham số
- Một số ví dụ: dựa trên lớp DaThuc
  - Xây dựng lớp Parabol
  - Xây dựng lớp đường Cung-Cầu Thị trường
  - Minh họa: Thi công tới đâu ➔ thử nghiệm tới đó!!
  - Vài gợi ý về bài tập Điện Thoại và Trạm phát song.

Một số ký hiệu UML

Quan hệ giữa hai lớp



- **Quan hệ Association**
  - **Lớp A có thuộc tính kiểu lớp B hoặc lớp B có thuộc tính kiểu lớp A**
  - **Ký hiệu UML**



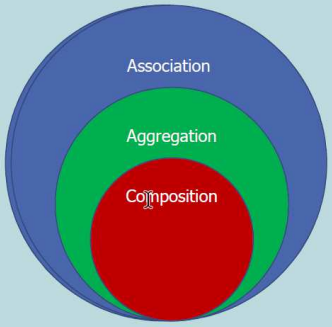
- **Quan hệ Composition**
  - **Lớp A và lớp B đã có quan hệ Association.**
  - **Lớp A có thuộc tính kiểu lớp B, nếu đối tượng a của lớp A bị hủy thì đối tượng b (thuộc tính của đối tượng a) của lớp B không thể tồn tại.**
  - **Ký hiệu UML**

- **Quan hệ Aggregation (has-a)**
  - **Lớp A và lớp B đã có quan hệ Association.**
  - **Lớp A có thuộc tính kiểu lớp B, nếu đối tượng a của lớp A bị hủy thì đối tượng b (thuộc tính của đối tượng a) của lớp B vẫn có thể tồn tại.**
  - **Ký hiệu UML**

Ví dụ

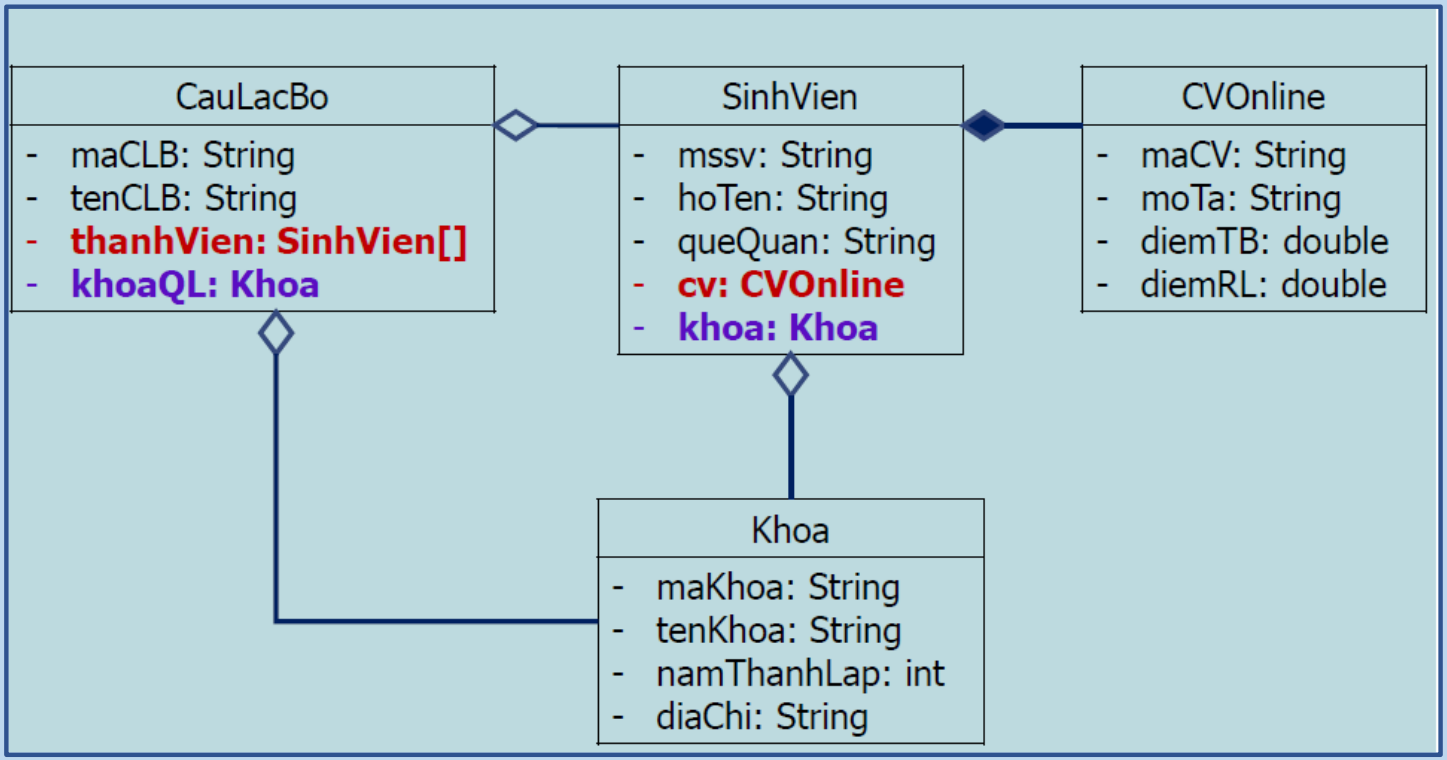
Sinh viên A thuộc khoa K quản lý và có thông tin CV trên mạng
Cậu lạc bộ CLB thuộc khoa K quản lý và có danh sách sinh viên tham gia

Quan hệ giữa hai lớp



Composition

Aggregation

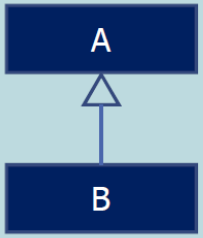


Xóa sinh viên A (nghỉ học): Phải xóa CV của A trong CVOnline (Composition).
Xóa Câu lạc bộ CLB: Xóa mà không cần phải điều chỉnh gì trong SinhVien và Khoa (Aggregation)

Một số ký hiệu UML

- **Quan hệ Dependency (uses-a)**
  - **Lớp A và lớp B không có quan hệ Association. Đối tượng kiểu lớp B có thể là đối số hoặc kết quả trả về hoặc biến cục bộ trong các phương thức của lớp A.**
  - **Ký hiệu UML**

- **Quan hệ kế thừa (is-a)**
  - **Lớp B kế thừa từ lớp A: lớp B là trường hợp đặc biệt của lớp A, lớp A là trường hợp tổng quát của lớp B.**
  - **Ký hiệu UML**




```
public class Diem
public class HinhTron

public long ViTriVoiHinhTron(Diem P) {….}
public Diem TamHinhTron()
  {
    return new Diem(this.x, this.y);
  }
```

```
public class HinhTron
public class HinhVanhKhan extends HinhTron

public class HinhEllipse extends HinhVanhKhan
```

Một số lưu ý về sự kế thừa trong Java

- Object là gốc của mọi lớp (class) cho dù phần định nghĩa lớp không ghi rõ rang kế thừa từ bất kỳ lớp nào. ➔ hữu ích khi dùng Object để "tập hợp" mọi đối tượng của các lớp khác nhau.
- Khai báo lớp kế thừa

  <phạm vi truy cập> [final] class <tên lớp con> extends <lớp cha>
- Chỉ cho phép kế thừa từ 1 lớp.
- Lớp kế thừa: thừa hưởng tất cả những gì public và protected trừ hàm xây dựng. Truy cập đến các thành phần lớp cha bằng super.<thành phần>
- Lớp kế thừa có các hàm xây dựng riêng và cần nên gọi hàm xây dựng lớp cha khi khởi tạo đối tượng (nhằm đảm bảo các vùng dữ liệu cần thiết của lớp cha được khởi tạo đúng)

Một số lưu ý về sự kế thừa trong Java

- Override: Cho phép định nghĩa lại phương thức có sẵn ở lớp cha để phù hợp với lớp con➔ phải trùng tên, trùng kết quả trả về, trùng dnah sách tham số.
- Nếu phương thức lớp cha khai báo là final ➔ không cho phép override phương thức này ở các lớp kế thừa.

	Overload	Override
Vị trí	Cùng trong 1 lớp	Lớp kế thừa
Tên	Cùng tên	Cùng tên
Số lượng	Có thể có nhiều hàm	Chỉ có 1
Tham số	Mỗi hàm có tham số khác nhau	Cùng danh sách tham số
Kiểu KQ	Cùng kiểu kết quả	Cùng kiểu kết quả

private: sống để bụng, chết mang theo
default: tài sản nổi
protected: tài sản chìm

	Lớp	Thuộc tính	Phương thức
final	Không cho kế thừa	Hằng	Không cho override

	private	default	protected	public
Cùng lớp				
Cùng gói				
Kế thừa				
Khác gói				

Cùng lớp: bản thân, Cùng gói: họ hàng
Kế thừa: cùng gói con ruột/khác gói: con nuôi
Khác gói+Không kế thừa: chả họ hàng chi!!

Gọi hàm khởi tạo/xây dựng lớp cha

- Lớp con phải có hàm khởi tạo riêng: dễ hiểu! do phải khởi tạo các vùng bổ sung them vào lớp cha.
- Hàm xây dựng lớp con cần gọi thi hành (ngay từ đầu) hàm khởi tạo lớp cha: dễ hiểu! việc khởi tạo các vùng kế thừa từ lớp cha sẽ cho hàm khởi tạo lớp cha thực hiện!! “con cái không quan tâm”
- Nếu không gọi thi hành hàm khởi tạo lớp cha? Java sẽ tự động gọi hàm khởi tạo không tham số của lớp cha ➔ khi xây dựng lớp thì NLT nên luôn xây dựng 1 hàm khởi tạo không tham số.
- Thứ tự gọi thi hành hàm khởi tạo ra sao?
  - Có thể viết 1 chương trình đơn giản để thử nghiệm.
  - Thử với các tình huống hàm khởi tạo lớp con gọi thi hành hàm khởi tạo lớp cha hoặc không.
  - Thử với tình huống lớp cha có/không có hàm khởi tạo không tham số

Gọi hàm khởi tạo/xây dựng lớp cha

```
class A {
public Diem d;
public A(){System.out.println("==>Ham khoi tao A()");}
public A(long x){System.out.println("Ham khoi tao A(long x)");}
}
```

```
class B extends A {
public B(){System.out.println("==>Ham khoi tao B()");}
public B(long x){System.out.println("Ham khoi tao B(long x)");}
}
```

```
class C extends B {
public C(){System.out.println("Ham khoi tao C()");}
public C(long a){ super(12L);
        System.out.println("Ham khoi tao C(long a)"); }
}
```

Thử nghiệm GoiHamKhoiTao.java với việc thay đổi để thử nghiệm nhiều trường hợp.

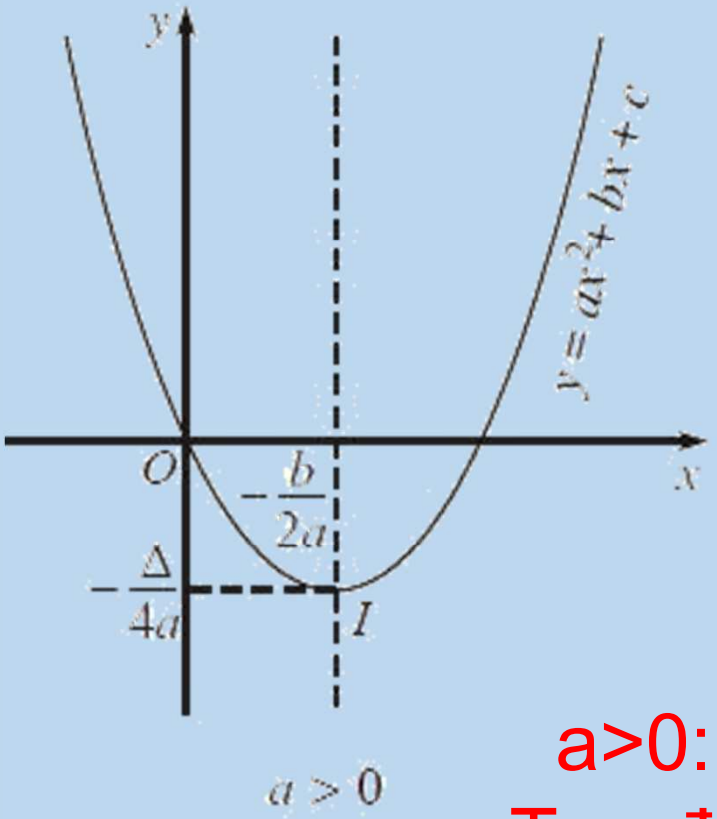
Lớp Điểm –nhắc lại do sẽ sử dụng

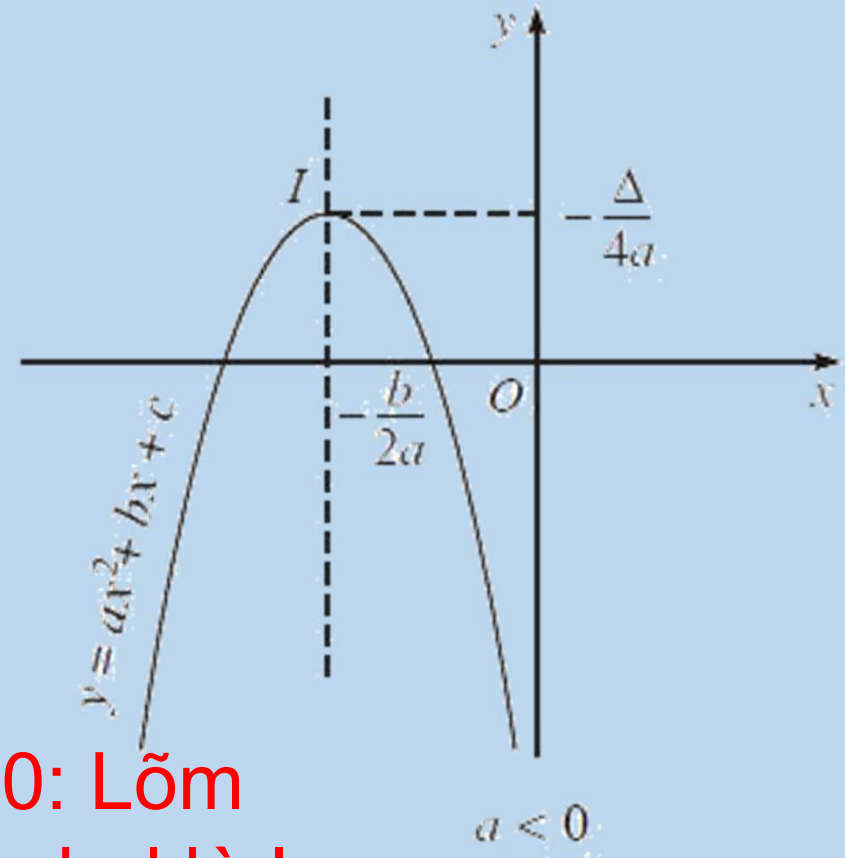
```
public class Diem {
   public double X;
   public double Y;
   public Diem() {X=Y=0;}
   public Diem(double x, double y){X=x;Y=y;}
   public Diem (Diem p){X=p.X; Y=p.Y;}
   public static double Dist(Diem p1, Diem p2)
   public double Dist(Diem p)
   public String toString()
}
```

Lớp DaThuc (bậc<=2)–nhắc lại do sẽ sử dụng

```
public class DaThuc
{
protected double a,b,c;
public DaThuc(){a=b=c=0;}
public DaThuc(double x, double y, double z){a=x;b=y;c=z;}
public double GiaTriDaThuc(double x)
public DaThuc CongDaThuc(DaThuc f)
public static DaThuc CongDaThuc (DaThuc f, DaThuc g)
public DaThuc NhanSoThuc (double k)
public String[] GiaiPhuongTrinh()
public DaThuc DaoHam()
public static DaThuc DaoHam(DaThuc f)
public String toString()
}
```

Parabol $y=ax^2 + bx + c$





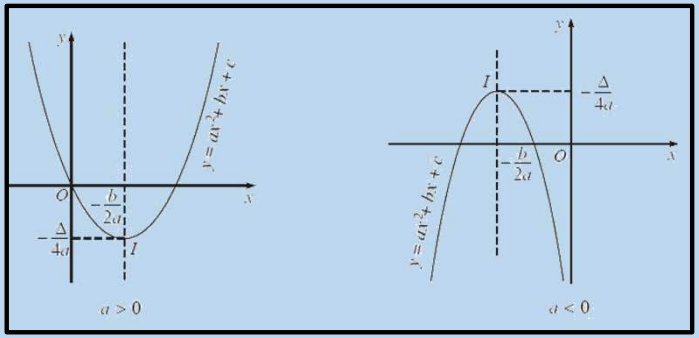
a>0: Lồi a<0: Lõm

Tọa độ đỉnh Parabol là I

Parabol $y=ax^2 + bx + c$

- Hướng của (P) phụ thuộc hệ số a (a<0 lõm, a>0 lồi)
- Đỉnh của (P) tại vị trí x=-b/(2a) và y= -Delta/(4a)
- (P) có thể cắt trục hoành, tiếp xúc hoặc không cắt trục hoành.
- Biến thiên của x

	-∞	**-b/(2a)**	**+∞**
a>0	Nghịch biến	Điểm cực tiểu	Đồng biến
a<0	Đồng biến	Điểm cực đại	Nghịch biến



Lớp Parabol

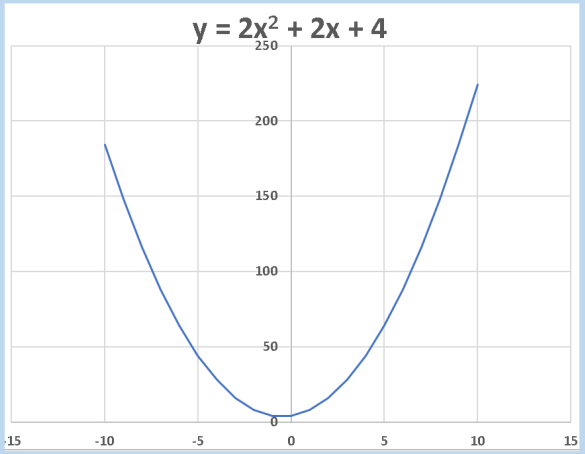
- Kế thừa từ lớp Đa thức.
- Hàm xây dựng: cần kiểm tra để hệ số a#0
- Bổ sung thêm vùng thông tin lưu lại giá trị Delta=b^2 – 4ac
- Phương thức toString: xuất các thông tin cơ bản của Parabol
- Phương thức cho biến khoảng biến thiên của Parabol.
- Phương thức cho biết vị trí 1 điểm D với Parabol (0: nằm trên P, -1: nằm trong P, +1: nằm ngoài P)
- Phương thức vẽ Parabol: Xuất ra danh sách các điểm trên mặt phẳng để vẽ Parabol.
- Phương thức để xác định giao điểm của 1 Parabol với Parabol khác/đường thẳng.

Lớp Parabol

```
public class Parabol extends DaThuc
{
    private double delta;
    private double Delta(){return b*b-4*a*c;}
    public Parabol()  {super(1,0,0);delta=Delta();}
    public Parabol(double a, double b, double c)
       {
        if (a==0) throw new ArithmeticException("He so a cua Parabol=0");
        this.a=a;this.b=b;this.c=c;delta=Delta();
       }
.....
```

Thử nghiệm với Parabol.java các hàm khởi tạo Parabol

Lớp Parabol




```
public String toString()  {
        String rs="Parabol có phuong trinh\n";
        rs+=super.toString()+"\n";
        rs+="Dinh Parabol la " +ToaDoDinhParabol().toString();
        return rs;
    }
public Diem ToaDoDinhParabol()  {
        return new Diem(-b/2/a,-delta/4/a);
    }
……
```

Thử nghiệm với Parabol.java với toString() và thông tin đỉnh parabol

Lớp Parabol

Xác định giá trị y ứng với x?

$$y=ax^2 + bx + c$$

➔ $y= f(x) = ax^2 + bx + c$

➔ $y_0 = f(x_0) + ax_0^2 + bx_0 + c$

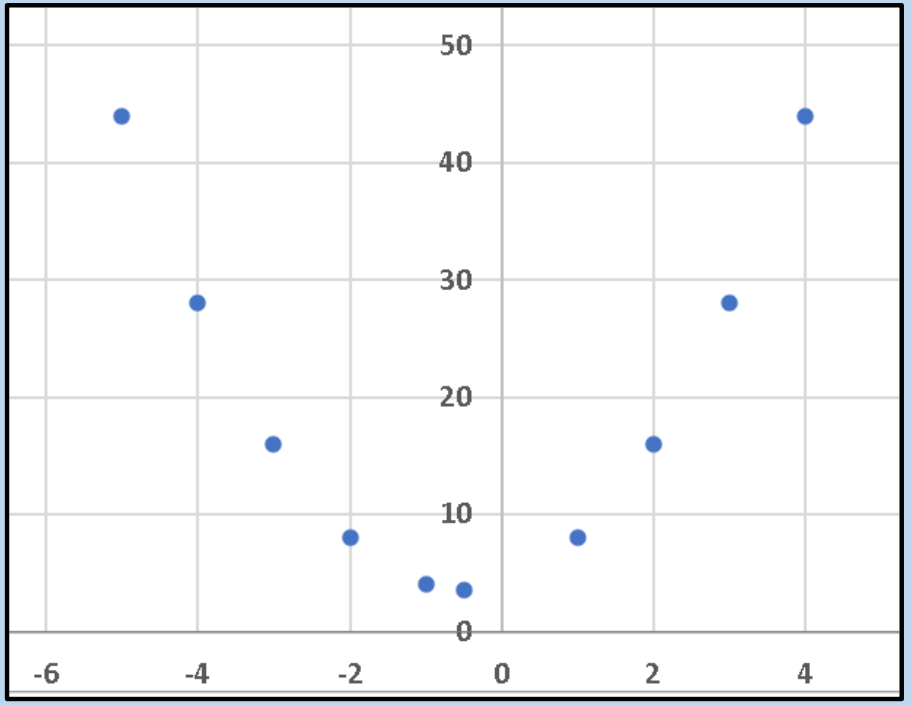
➔Tìm giá trị đa thức bậc 2 ➔ Kế thừa phương thức của lớp DaThucBac2

```
public double GiaTriDaThuc(double x)
{
    return a*x*x+b*x+c;
}
```

Thử nghiệm với Parabol.java với việc tính giá trị y khi biết x

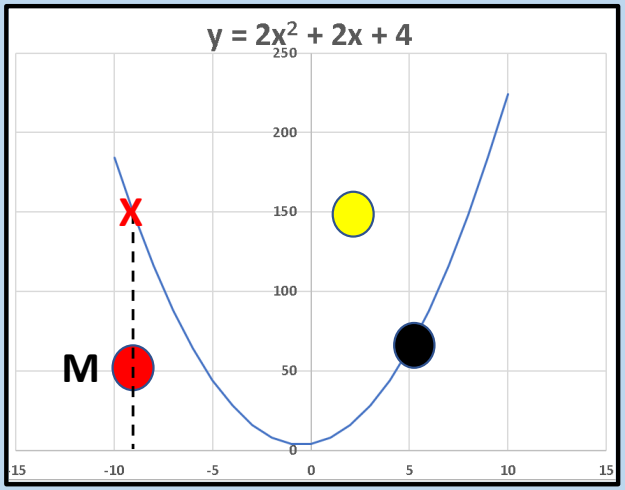
Lớp Parabol

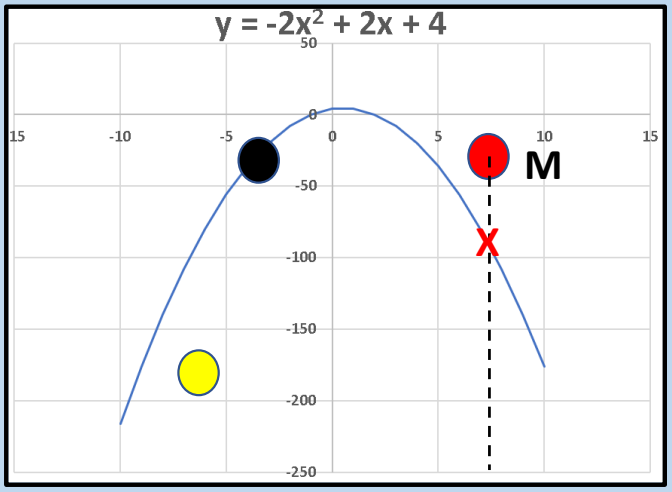
```
static private double kc=0.5;
 static private int sodiem=10;
public Diem[] VeParabol()
 {
  Diem[] dsdiem= new Diem[sodiem];
  Diem dinh=ToaDoDinhParabol();
  double left=dinh.X-(sodiem/2)*kc;
  for(int i=0;i<sodiem;i++)
    {
     Diem x= new Diem (left,this.GiaTriDaThuc(left));
     dsdiem[i]=x;
     left+=kc;
    }
  return dsdiem;
 }
```



Thử nghiệm với Parabol.java với VeParabol()

Lớp Parabol



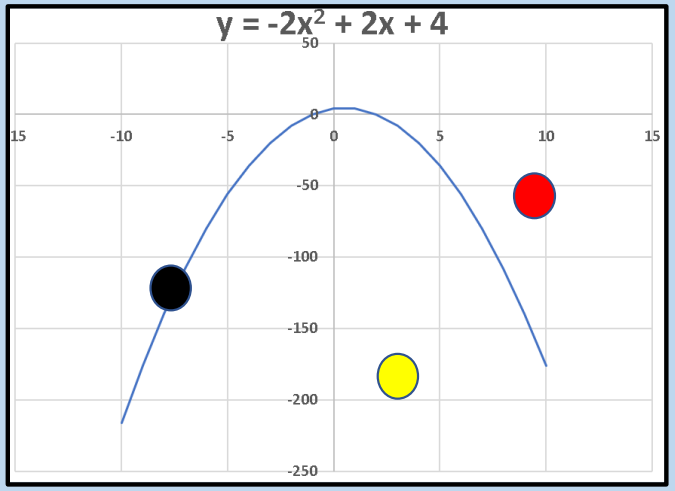


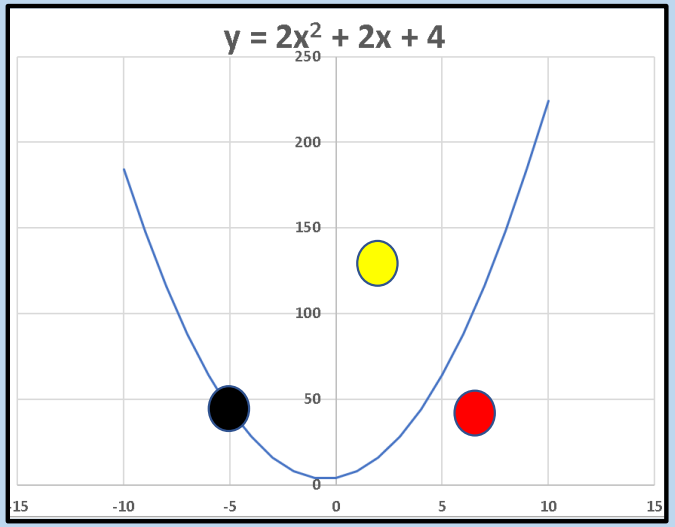
- Vị trí 1 điểm M so với Parabol ➔
  phụ thuộc hệ số a
  so sánh $ax_M^2 + bx_M + c$ *với* y_M
- Qui ước:
  Trả về 0: nếu nằm trên P điểm màu đen
  Trả về -1: nếu nằm trong P điểm màu vàng
  Trả về +1: nếu nằm ngoài P điểm màu đỏ

Lớp Parabol

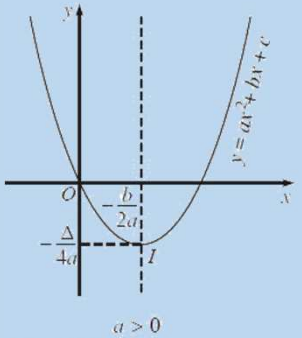
```
public long ViTriDiemVoiParabol(Diem P)
 {
 double Y=this.GiaTriDaThuc(P.X);
 if (Y==P.Y) return 0;
 if (a>0&&P.Y>Y) return -1;
 if (a<0&&P.Y<Y) return -1;
 return 1;
  }
```

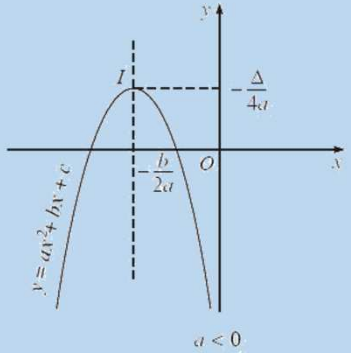
Thử nghiệm với Parabol.java với ViTriDiemVoiParabol





Lớp Parabol






```
public String BienThien(double left, double right)  {
 Diem dinh=ToaDoDinhParabol();
 if (a>0)  {
  if (dinh.X<=left) return "Nghich bien tren doan ["+left+","+right+"]";
  if (dinh.X>=right)  return "Dong bien tren doan ["+left+","+right+"]";
  return "Nghich bien tren doan ["+left+","+dinh.X+"] va Dong bien tren doan ["+dinh.X+","+right+"]";
            }
 if (a<0)
          {
  if (dinh.X<=left) return "Dong bien tren doan ["+left+","+right+"]";
  if (dinh.X>=right)  return "Nghich bien tren doan ["+left+","+right+"]";
  return "Dong bien tren doan ["+left+","+dinh.X+"] va Nghich bien tren doan ["+dinh.X+","+right+"]";
        }
}
```

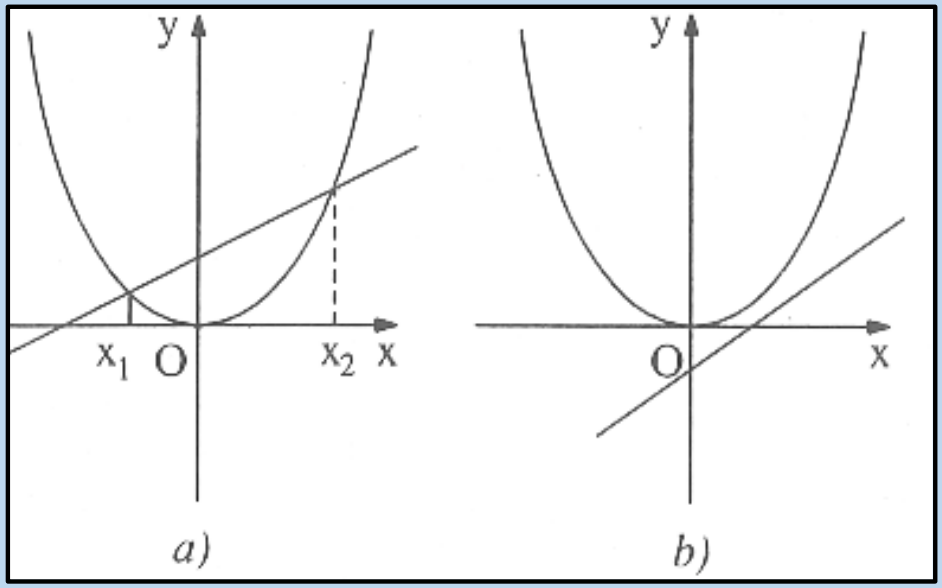
Thử nghiệm với Parabol.java với phương thức BienThien

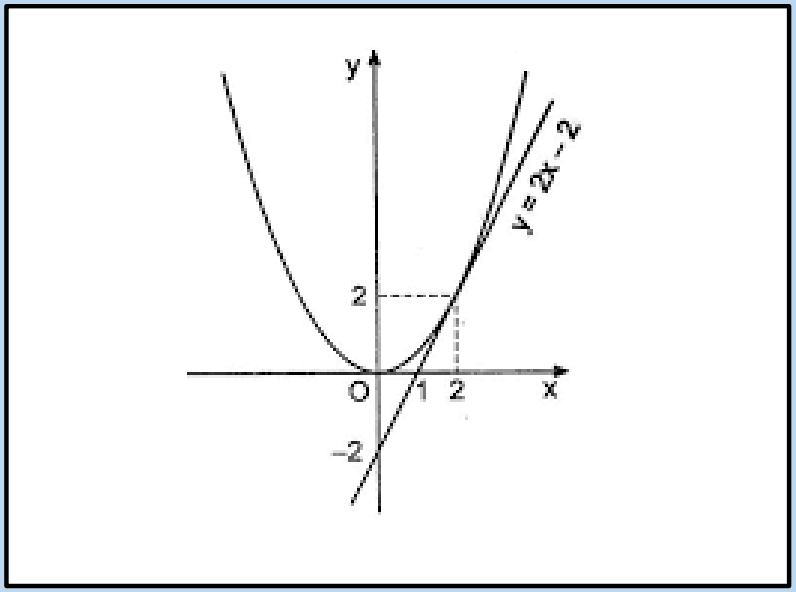
Lớp Parabol

```
public DaThuc CongDaThuc(DaThuc f)
public static DaThuc CongDaThuc (DaThuc f, DaThuc g)
public DaThuc NhanSoThuc (double k)
public String[] GiaiPhuongTrinh()
```

- Giao điểm Parabol với đường thẳng D mx+n:
- Thực chất là giải phương trình $ax^2 + bx + c = mx + n$

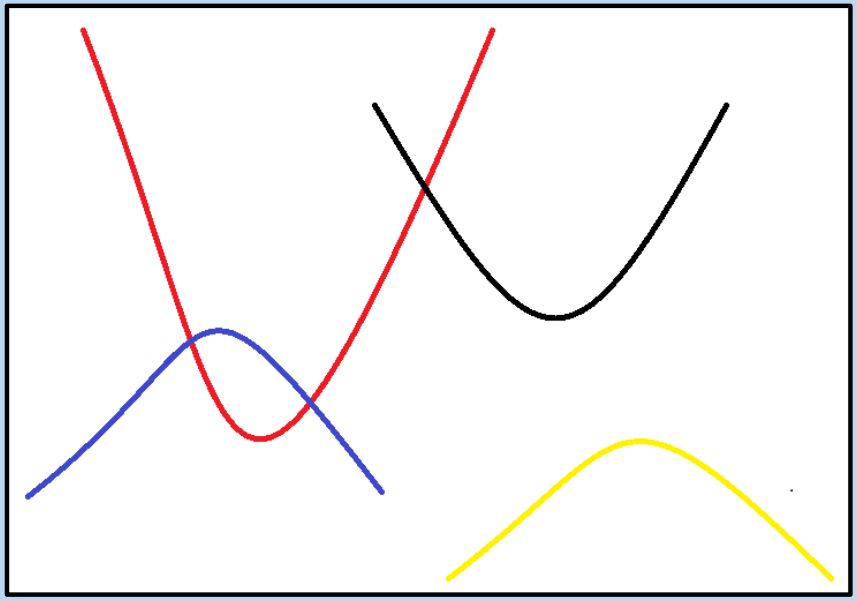
➔ Kế thừa phương thức Giải Phương trình từ DaThuc





Lớp Parabol

```
public DaThuc CongDaThuc(DaThuc f)
public static DaThuc CongDaThuc (DaThuc f, DaThuc g)
public DaThuc NhanSoThuc (double k)
public String[] GiaiPhuongTrinh()
```



- Vị trí tương đối của 2 Parabol: không cắt nhau, cắt nhau, trùng nhau
  - Parabol hiện tại: màu đỏ.
  - Parabol màu đen: chỉ giao nhau tại 1 điểm.
  - Parabol màu xanh: cắt nhau tại 2 điểm.
  - Parabol màu vàng: không cắt nhau.
- Thực chất: Giải phương trình P=P1

➔Kế thừa từ DaThuc

P=P1

➔ P- P1=0

➔ P + (P1 x -1) =0

➔ R= Cộng P với Q (Q=P1 nhân với số thực -1)

➔Giải R=0

Lớp Parabol

```
public DaThuc CongDaThuc(DaThuc f)
public static DaThuc CongDaThuc (DaThuc f, DaThuc g)
public DaThuc NhanSoThuc (double k)
public String[] GiaiPhuongTrinh()
```

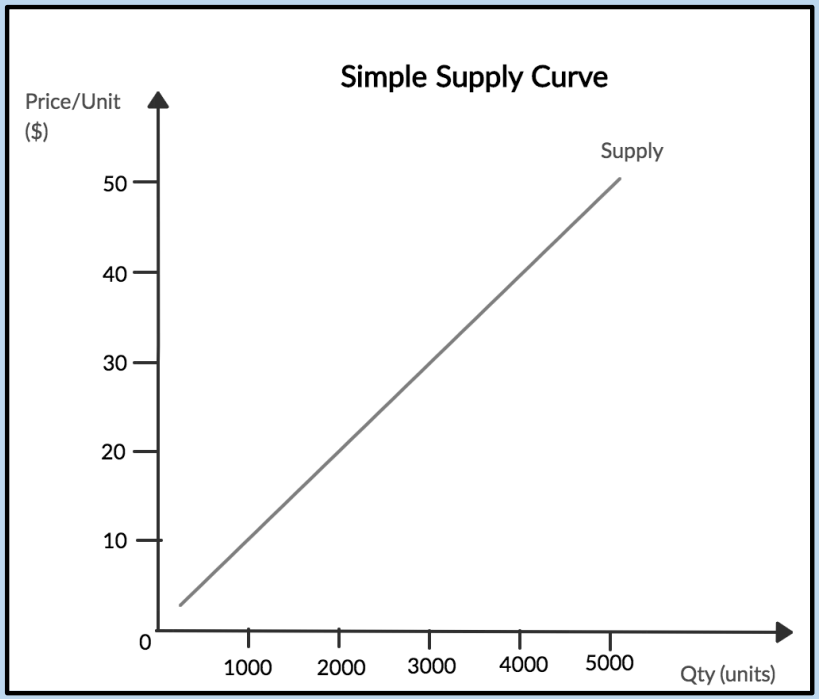
```
public void GiaoDiemVoiParabol(DaThuc Q)
  {
   String[ ] kq=this.CongDaThuc(Q.NhanSoThuc(-1)).GiaiPhuongTrinh();
   System.out.println("Giao diem tai:");
   for(String s:kq)System.out.println(s);
  }
```

Parabol kế thừa từ DaThuc ➔ có các phương thức CongDaThuc, NhanSoThuc, GiaiPhuongTrinh

Thử nghiệm với Parabol.java với GiaoDiemVoiParabol

Kinh tế vi mô: Đường cung và đường cầu

- Lý thuyết cơ bản về cung và cầu 1 mặt hàng trên thị trường.
- Dùng đồ thị để mô tả quan hệ giữa giá bán và số lượng mặt hàng.
- Trục tung: giá bán mặt hang (Price – P)
- Trục hoành: số lượng mặt hang (Quantity – Q)



- Cung ứng (Supply-S): Nếu giá bán cao thì sẽ sản xuất (cung) nhiều, ngược lại nếu giá bán thấp sẽ sản xuất ít.
  - ➔ Đồng biến!!!
  - ➔ Đơn giản hóa quan hệ bằng đường thẳng ax+b với a>0

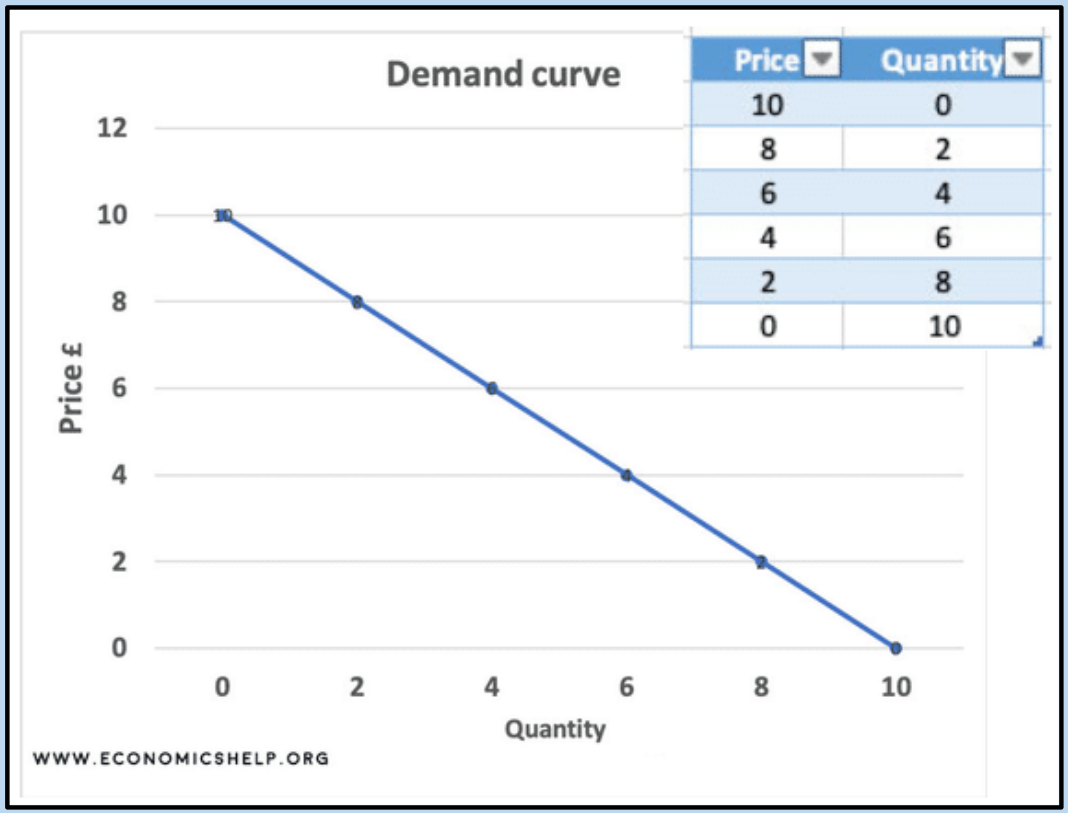
Kinh tế vi mô: Đường cung và đường cầu

- Nhu cầu (Demand-S): Nếu giá bán cao thì lượng mua thấp, ngược lại nếu giá bán thấp thì lượng mua nhiều.

  ➔ Nghịch biến!!!

  ➔ Đơn giản hóa quan hệ bằng đường thẳng ax+b

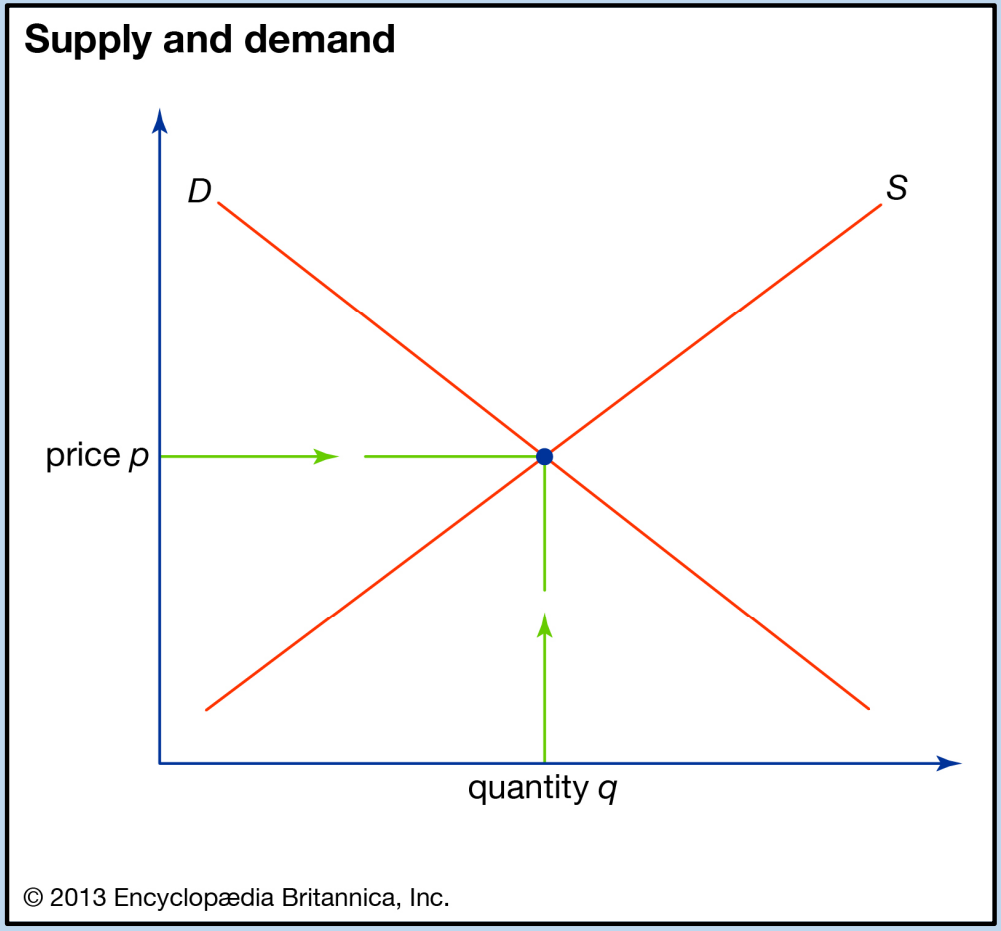
  với a<0



Price	Quantity
10	0
8	2
6	4
4	6
2	8
0	10

Điểm cân bằng cung-cầu

- Đưa 2 phương trình đường cung và cầu của 1 mặt hàng lên đồ thị.
- Giao điểm của 2 đường gọi là điểm cân bằng thị trường:
  - Tại đó: Số lượng cung cấp và giá bán hợp lý của thị trường.
  - Trượt điểm cân bằng theo D (Nhu cầu) ➔ giá cả sẽ tăng giảm/lượng cung ứng sẽ giảm/tăng.
  - Trượt điểm cân bằng theo S (Cung ứng) ➔ giá cả sẽ tăng giảm/lượng cung ứng sẽ tăng/giảm.



Lớp CungCau

- Kế thừa từ lớp DaThuc.
- DaThuc: biểu diễn các đa thức $ax^2 + bx + c$
- Hệ số a phải là 0.
- Đường cầu và cung có phương trình biểu diễn là ax + b

➔ a đường cung cầu là b của DaThuc, b đường cung cầu là c của DaThuc

- Đường cung: a>0
- Đường cầu : a<0

➔ Xây dựng lớp CungCau và bổ sung them các vùng thông tin, phương thức cần thiết.

Lớp CungCau

```
public class CungCau extends DaThuc {
 String mathang;
 boolean cung;
 CungCau(boolean laduongcung, String tenMH)
            {  super(); cung=laduongcung;mathang=tenMH; }
 CungCau (boolean laduongcung,String tenMH, double a, double b){
     super(0,a,b);
     if (laduongcung&&a<0)throw new ArithmeticException("He so a cua
duong cung phai>0");
     if (!laduongcung&&a>0)throw new ArithmeticException("He so a cua
duong cau phai<0");
    cung=laduongcung;mathang=tenMH;
   }
```

Thử nghiệm với CungCau.java khởi tạo các đường cung và cầu

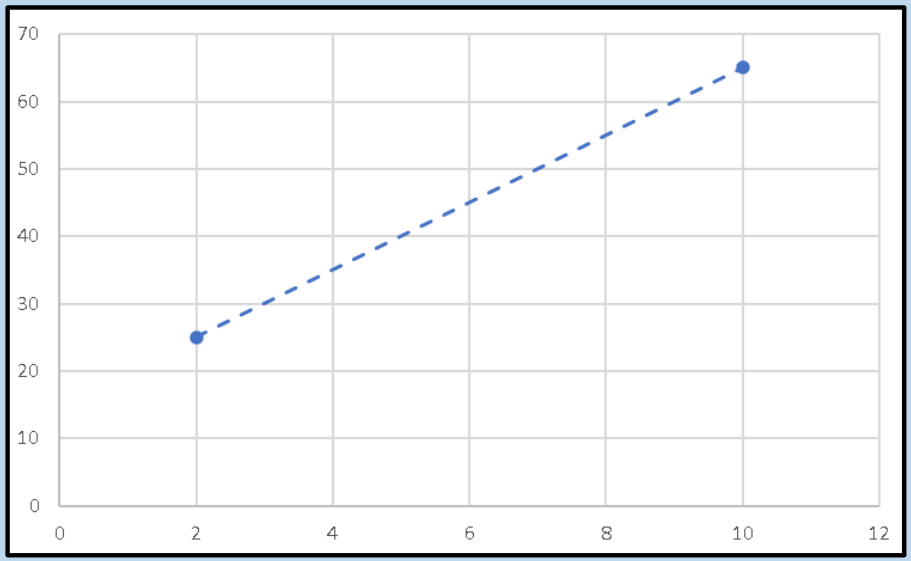
Lớp CungCau

Thử nghiệm với CungCau.java khởi tạo bằng cặp điểm

- Khởi tạo đường cung/cầu dựa vào cặp tọa độ (Q,P)

```
CungCau(String tenMH,double Q1, double P1, double Q2, double P2)
 {
 double b= (P2-P1)/(Q2-Q1);
 double c= P1-(P2-P1)*Q1/(Q2-Q1);
 if (b>0)cung=true; else cung=false;
 mathang=tenMH;
 this.b=b;
 this.c=c;
 }
```

Q	P
2	25
10	65



Lớp CungCau

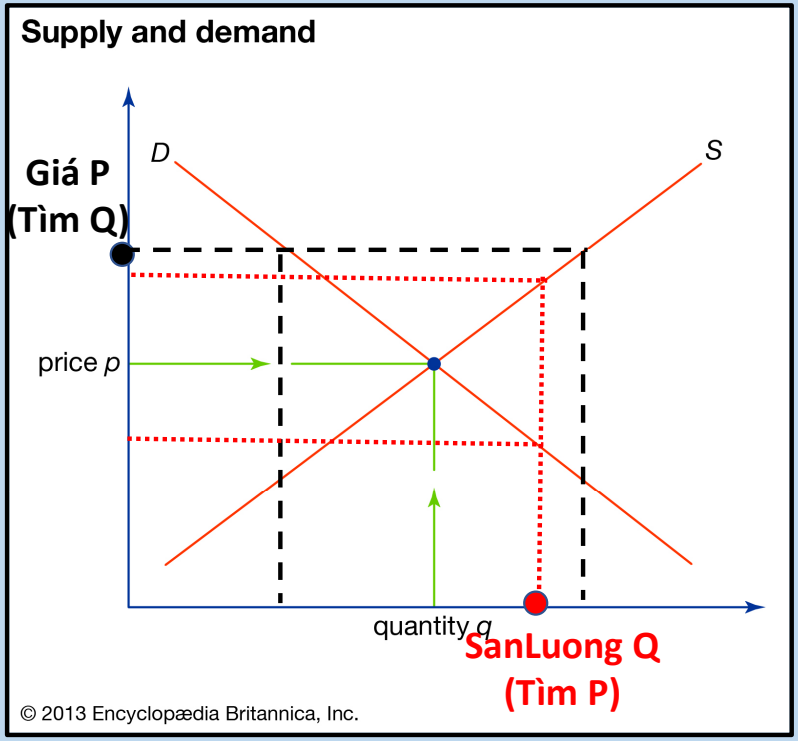
- Tìm giá bán khi biết số lượng

➔ thay thế số lượng Q vào phương trình

```
public double Gia(double SanLuong)
    {  return GiaTriDaThuc(SanLuong); }
```

- Tìm số lượng khi biết giá bán

```
public double SanLuong(double Gia)
    {  return (Gia-c)/b;  }
```



Có thể vận dụng GiaiPhuongTrinh
Tìm x để $0x^2 + bx + c = Gia$
➔$0x^2 + bx + (c-Gia)= 0$
➔Da Thuc d= new DaThuc(0,b,c-Gia);
➔d.GiaiPhuongTrinh() ➔ Kết quả
Phức tạp hơn !!!

Thử nghiệm với CungCau.java
để tìm giá khi biết sản lượng cung/cầu
và tìm sản lượng khi biết giá mua/bán

Lớp CungCau

- Tìm điểm cân bằng cung/cầu
- Vị trí 2 đường cắt nhau

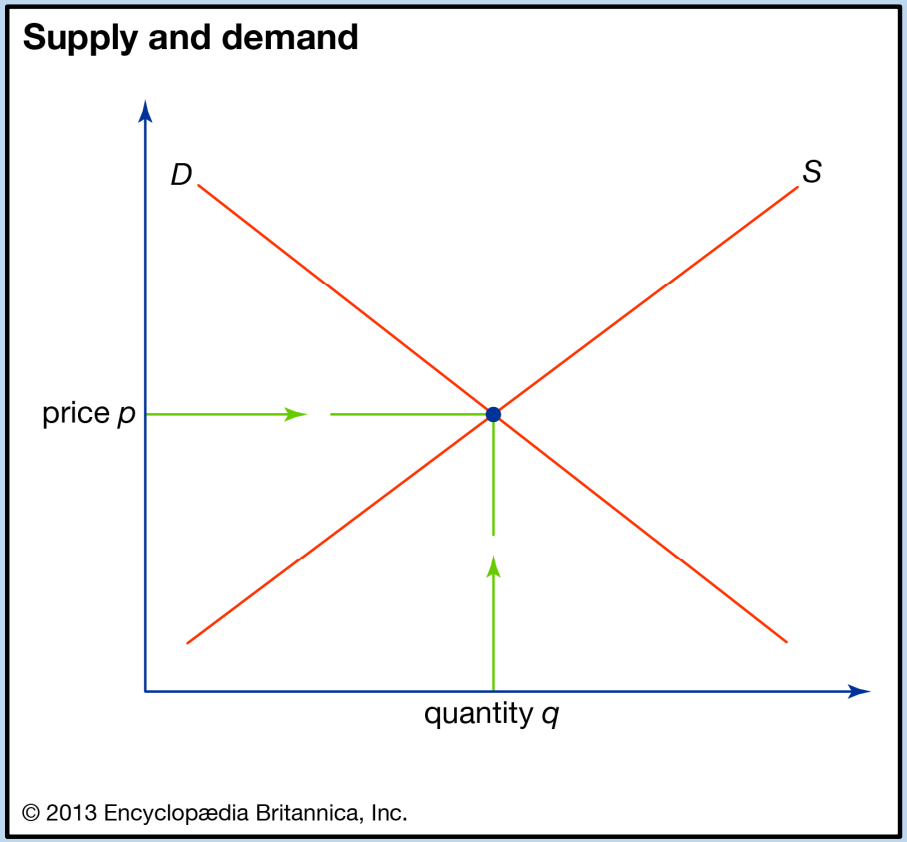
➔ (D)=(S)

➔ Giải phương trình (D)-(S)=0

2 đường thẳng khác nhau cắt nhau tại 1 điểm

Sử dụng

- Nhân đa thức (S) với số thực -1 : (-S)
- Cộng đa thức (D) với (–S): U
- Giải phương trình U= 0 ➔ sản lượng cân bằng Q
- Thay Q vào (D) để có giá cân bằng P



Lớp CungCau

```
public void DiemCanBang(CungCau d)
 {
  DaThuc dt= this.CongDaThuc(d.NhanSoThuc(-1));
  String[ ] kq= dt.GiaiPhuongTrinh();
  double x= Double.parseDouble(kq[0]);
  System.out.println("Diem can bang tai Q1= "+x+" và P= "+
                                                this.GiaTriDaThuc(x));
 }
```

Thử nghiệm với CungCau.java
để tìm điểm cân bằng

Bài thực hành

- Có 1 tuần nghỉ online!!
- Thực hành các bài tập Lab 1,2 và 3 (đã post lên LMS)
- Bài thực hành Điện Thoại và Trạm Phát Sóng
  - Làm nhiều tuần và với nhiều mức độ khác nhau.
  - Xây dựng lớp Điện thoại kế thừa Điểm
  - Xây dựng lớp Trạm phát song kế thừa Hình tròn.
  - Các thông tin cần thiết cho 2 lớp (tùy ý anh chị thiết kế và tổ chức)
  - Các phương thức cần thiết cho 2 lớp (tùy ý anh chị)
  - Tự đề xuất giao thức kết nối, ví dụ: Trạm phát sóng thụ động, điện thoại có lưu danh sách các trạm phát song sau đó sẽ chọn lựa trạm phát tốt nhất!!
- Phát triển trong các tuần sau:
  - Độ mạnh sóng giảm dần theo bán kính phát song.
  - Mô phỏng các điện thoại di chuyển và thay đổi trạm phát sóng phục vụ.
  - Mô phỏng thêm các tình huống trạm phát sóng (tắt, tăng công suất, …)

class DienThoai – chỉ là gợi ý

```
public class DienThoai extends Diem {
 public String SoDienThoai;
 public boolean HoatDong;
 public String MaTram;
 public DienThoai()
 public DienThoai(String sdt, boolean tt, double x, double y, Str
 public DienThoai(DienThoai d)
 public void BatDienThoai()
 public String toString()
 public boolean CoThePhucVu (TramPhatSong Tram)
 public void TimTram(ArrayList<TramPhatSong> ds)
 .......
}
```

HoatDong: TRUE nếu điện thoại đang bật, FALSE nếu điện thoại đang tắt
Nếu điện thoại đang có 1 trạm phục vụ thì sẽ lưu mã trạm ấy vào vùng MaTram, nếu ngoài vùng phủ song/tắt → null/rỗng
BatDienThoai(): Mô phỏng hành động bật điện thoại ON
TimTram: duyệt từng trạm phát song, với mỗi trạm sẽ gọi CoThePhucVu để xem trạm phát song có thể phục vụ cho điện thoại không (điện thoại nằm trong vùng phủ song và số lượng điện thoại phục vụ < Max). Sau đó sẽ chọn ra trạm tốt nhất trong số các trạm có khả năng phục vụ

class TramPhatSong – chỉ là gợi ý

```
public class TramPhatSong extends HinhTron {
 public String MaTram;
 public String DiaChi;
 public double DoManh;
 public long SoLuongKetNoiMax;
 public ArrayList<DienThoai> danhsach;
 public TramPhatSong(){ }
 public TramPhatSong(String ma, String dc,double domanh, long som
 .......
}
```

Mã trạm và địa chỉ trạm: Tùy anh chị tổ chức.
SoLuongKetNoiMax: số lượng điện thoại tối đa mà trạm có thể phục vụ cùng 1 lúc
danhsach: là danh sách các số điện thoại đang phục vụ, sẽ cập nhật khi có điện thoại không còn được phục vụ/điện thoại bắt đầu được phục vụ.
Bán kính phát sóng là R (thừa kế từ HinhTron)
Tạm thời độ mạnh của sóng là như nhau trong vùng phủ song.
Tương lai: Độ mạnh tại 1 điểm trong vùng phát song phụ thuộc vào 1 hàm (sử dụng lớp DaThuc!!!)

class Main với hàm main để thử nghiệm

```
import java.util.ArrayList;
public class Main
{
 static ArrayList<TramPhatSong> DSTram= new ArrayList<TramPhatSon
 static ArrayList<DienThoai> DSDienThoai= new ArrayList<DienThoai
 static
 {
     //Khởi tạo 1 số trạm phát sóng tại đây.
 }
 public static void main(String[] args)
   {
    //Khởi tạo các điện thoại để thử nghiệm các phương thức
   }

}
```

Tương lai:
Cần xem them các phương thức ngẫu nhiên (**random**) để mô phỏng toạ độ của điện thoại sẽ thay đổi do người dung di chuyển ➔ sẽ cập nhật thông tin trong điện thoại và trạm phát song, ngẫu nhiên thêm điện thoại, ngẫu nhiên tắt điện thoại.
Cần xem thêm **Thread:** mô phỏng tự động chứ không thủ công!! Tạo 2 công việc được làm cùng lúc (xử lý song song), 1 công việc là sẽ thay đổi ngẫu nhiên vị trí các điện thoại và công việc kia là tự động cập nhật thông tin của điện thoại và trạm phát sóng.