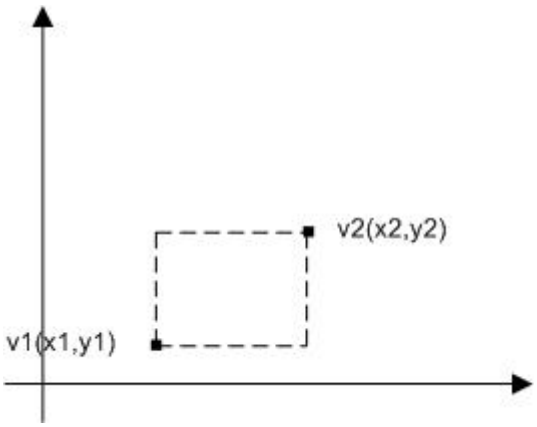
Este exercício tem como contexto uma hierarquia de polígonos semelhante à vista anteriormente, mas restrita a apenas duas classes: uma chamada **Retangulo** e outra chamada **Quadrado**, que é subclasse da primeira.

Para criarmos um retângulo cujos lados sejam paralelos aos eixos dos **x** e dos **y** do plano cartesiano, é necessário fornecermos apenas dois pontos que não definam retas paralelas a tais eixos. Dessa forma, o segmento de reta formado pelos dois pontos será uma das diagonais do retângulo. A figura abaixo ilustra o que acabou de ser dito:



Logo, é fácil construir um algoritmo que receba dois pontos, verifique se tais pontos permitem construir um retângulo como o descrito acima e, além disso, verifique se o retângulo é ou não um quadrado. Entretanto, este algoritmo não deve ser implementado como um construtor da classe **Retangulo** ou da classe **Quadrado**. Os motivos são dois:

1. Algum código, de alguma função, teria que determinar, a priori, se os pontos formam um quadrado ou um retângulo, para então criar o objeto correto. Ora, não é essa exatamente a decisão que gostaríamos de delegar aos construtores das classes?

2. Caso os pontos definam uma reta paralela a um dos eixos o retângulo não seria válido. Nesse caso teríamos que abortar a criação gerando uma exceção dentro do construtor. Isso aumentaria a complexidade do código da função que deseja criar um retângulo, pois ela teria que tratar a exceção.

Uma boa solução para esse problema é uma técnica conhecida na literatura por fábrica (*factory*). Uma fábrica é uma classe cujo objetivo é reunir algoritmos que implementem critérios complexos para a criação de objetos. No nosso caso, como desejamos criar objetos de apenas duas classes distintas por meio de um algoritmo relativamente

simples, uma boa solução seria delegar a um método da própria classe retângulo a tarefa de criar as instâncias. Como não faria o menor sentido instanciar um retângulo para tal, já que esse é exatamente o problema que desejamos resolver, a solução seria definir um método estático da própria classe **Retangulo** para produzir tais instâncias.

Logo, a sua primeira tarefa será criar um método para a criação de objetos da classe retângulo, cuja assinatura deve ser a seguinte:

```
public static Retangulo create(Ponto p1,Ponto p3)
```

Desse momento em diante, todas as vezes que quisermos criar um retângulo chamaremos o método acima passando dois pontos como parâmetros. Caso uma instância possa ser criada o método retornará uma referência para a mesma, caso contrário ele retornará `null` (nenhum objeto foi criado). Note que o método acima poderá criar uma instância de um quadrado, caso os argumentos definam um. Veja também que o seu código não precisará tomar conhecimento desse detalhe. **Isso é o polimorfismo em ação!!!**

A outras classes e os outros métodos do exercício são os seguintes:

```
package poligono;
public class Ponto {
	private double x,y;

	public Ponto(double x,double y) {
		// complete o código

	}

	public double getX() {
		// complete o código
	}

	public double getY() {
		// complete o código
	}

	public double dist(Ponto p) {
		// complete o código
	}
}


package poligono;
public class Retangulo {
	protected Ponto[] vert;

	public static Retangulo create(Ponto p1,Ponto p3) {
```

```
            // complete o código
      }

      Retangulo(Ponto p1,Ponto p2,Ponto p3,Ponto p4) {
            // complete o código
      }

      public double perimetro() {
            // complete o código
      }

      public void tipo() {
            System.out.println("Sou um Retângulo");
      }
}

package poligono;
public class Quadrado extends Retangulo {
      Quadrado(Ponto p1,Ponto p2,Ponto p3,Ponto p4) {
            // complete o código
      }

      public double perimetro() {
            // complete o código
      }

      public void tipo() {
            System.out.println("Sou um Quadrado");
      }
}
```

O seu programa deve ler dois pontos do teclado e tentar criar um retângulo por meio da fábrica. Caso ele tenha sido criado você deve exibir o seu tipo (método `tipo()`) e perímetro. Caso contrário, você deve exibir a mensagem **Pontos inválidos**.