Announcements/Follow-ups

- P7 is posted, due Friday August 2 at 11pm
  - No late period
  - You can choose either FishPond or YearGoogol
  - Secret Tests
- Follow-ups
  - Bidirectional example fix: remove guava from build-path
  - StaticGenerics example
  - Finish Wednesday's slides
  - ArrayList copy constructor and generic wild-card <?>
  - Year Googol background

Inheritance and generics

- Extends can be used with type Parameters
  - LineUp<A extends Athlete>
    - Now A can't be any type, only Athlete and its sub-types
  - MyGeneric<T **extends** MyInterface>
    - "Extends" was chosen as the keyword in type-parameterization, even for interfaces
    - When your class implements an interface, you write "implements"
    - But when your interface is derived from a super-interface, you write "extends"
    - ShapeList example

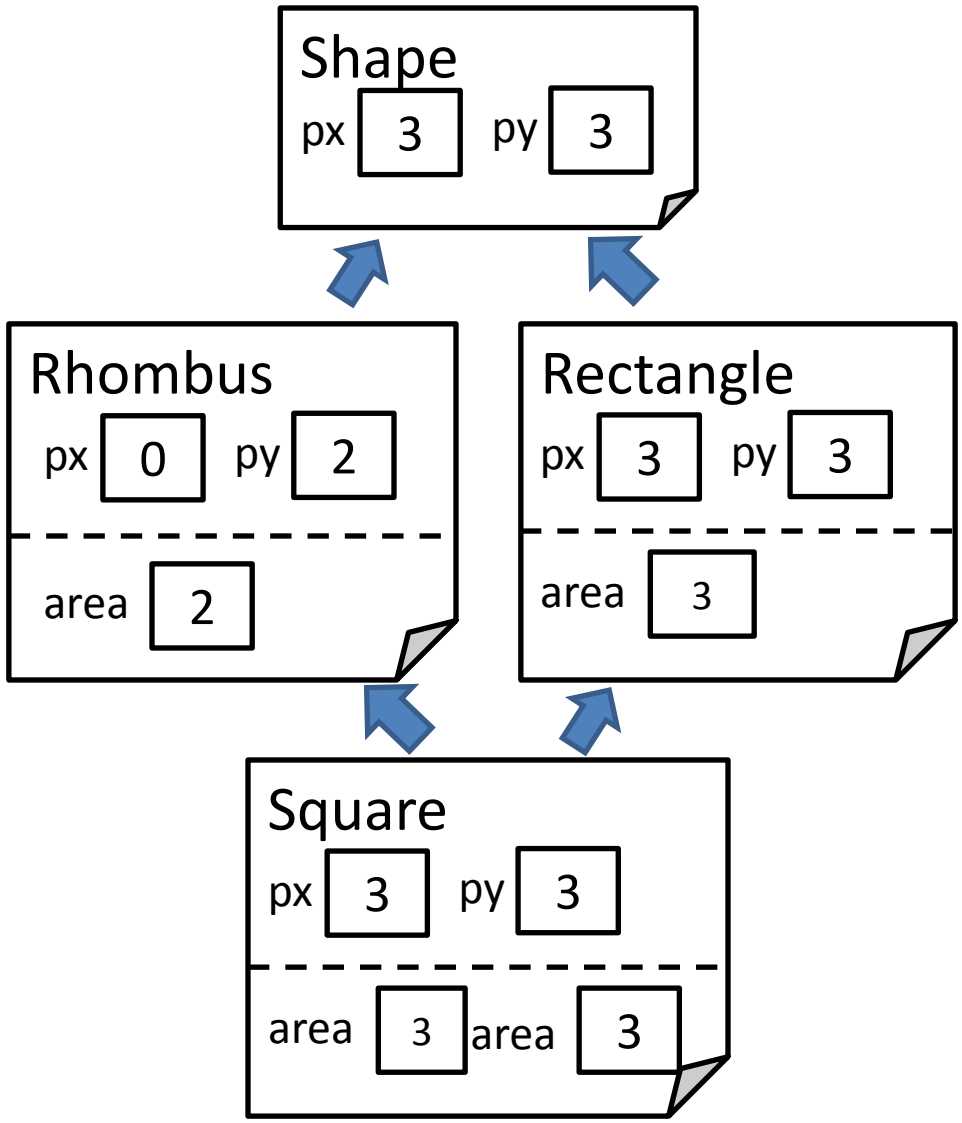
Inheritance and generics

- ArrayList copy constructor and <?>
  - ShapeList again
- Arrays are covariant, collections are not
  - ArraysAndGenerics example

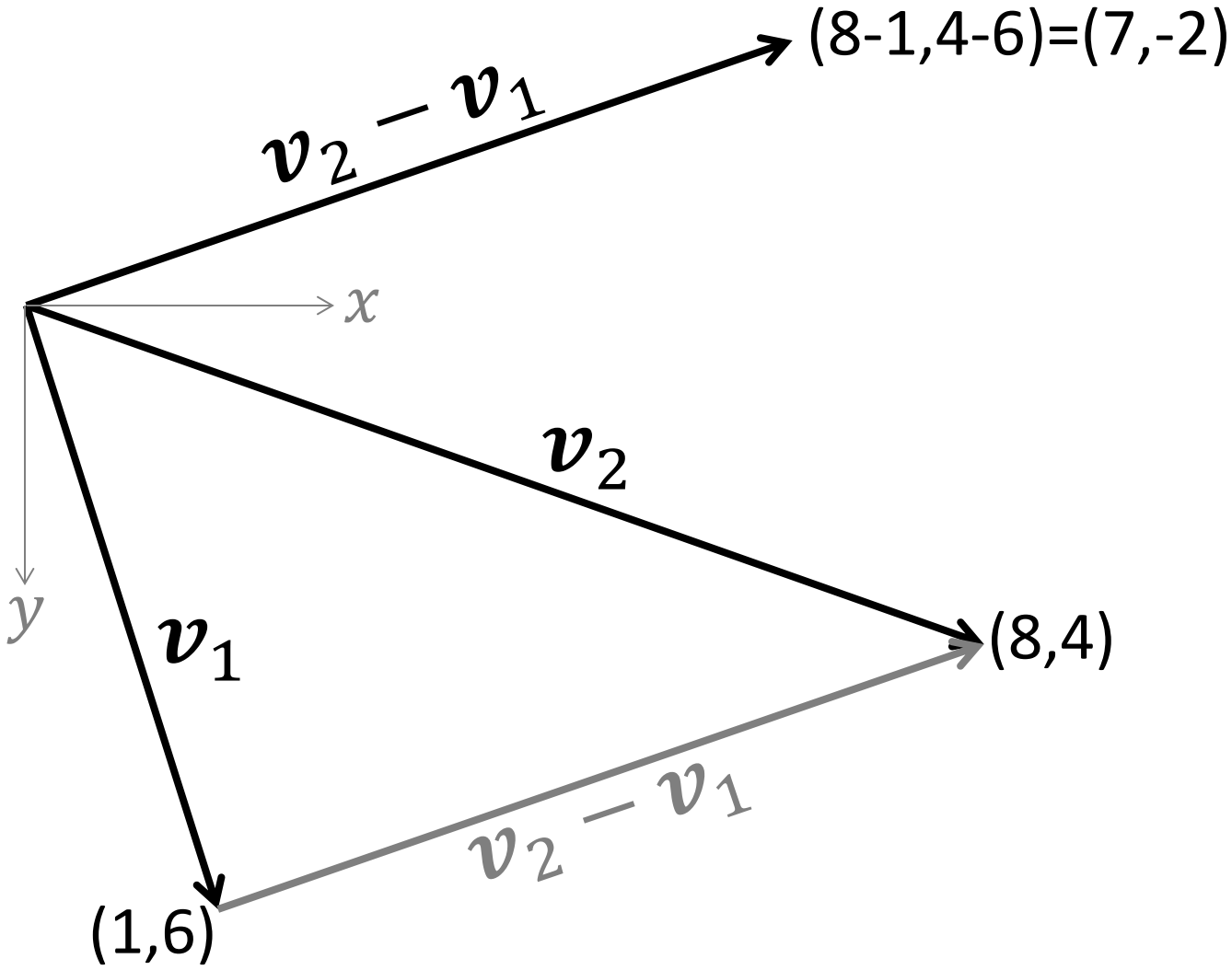
Multiple Inheritance

```
class Square
        extends
        Rhombus,
        Rectangle
{…}
```

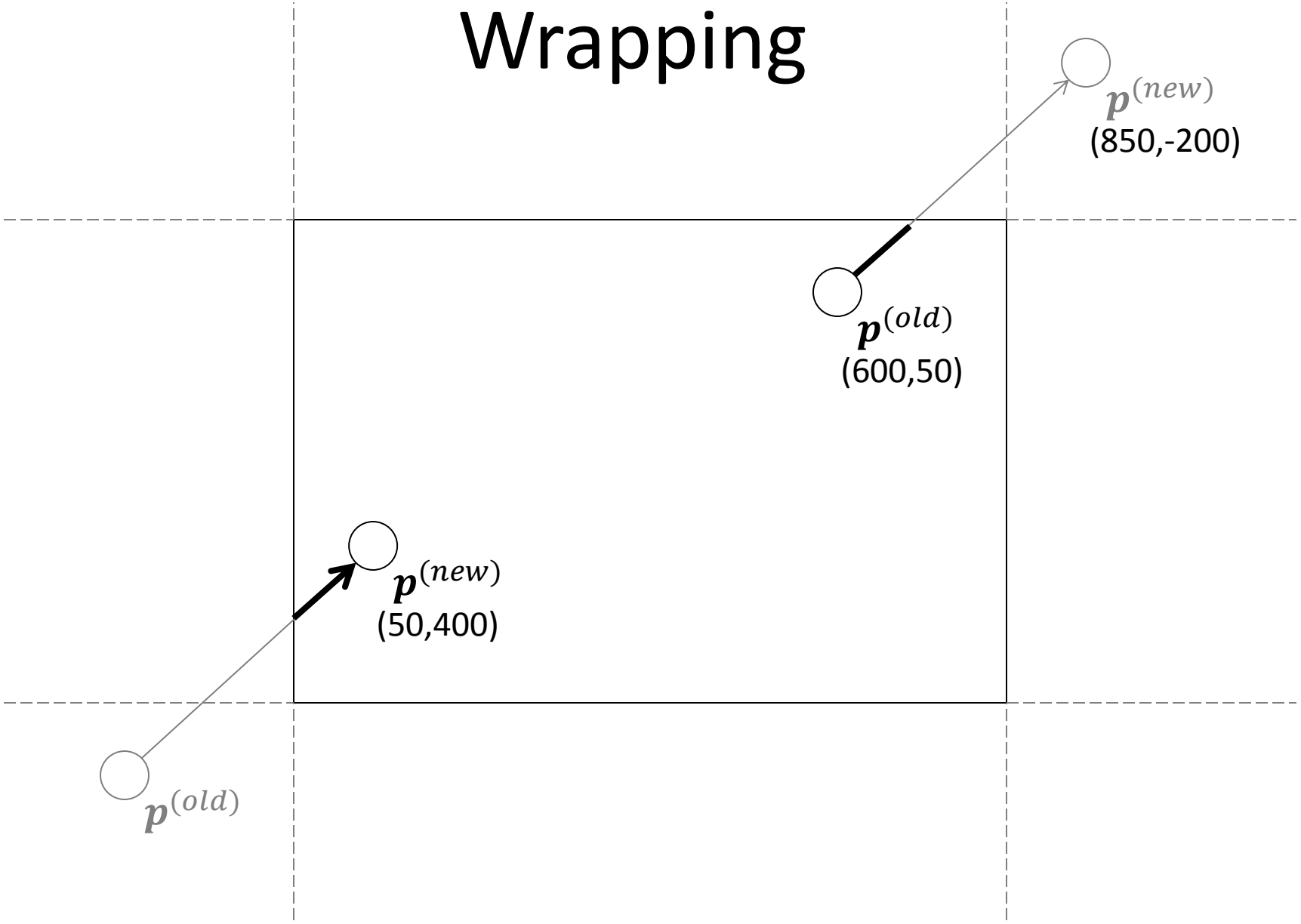
- Issues?
- Supported in Java for interfaces, but not classes
- Supported for classes in other languages, e.g. C++



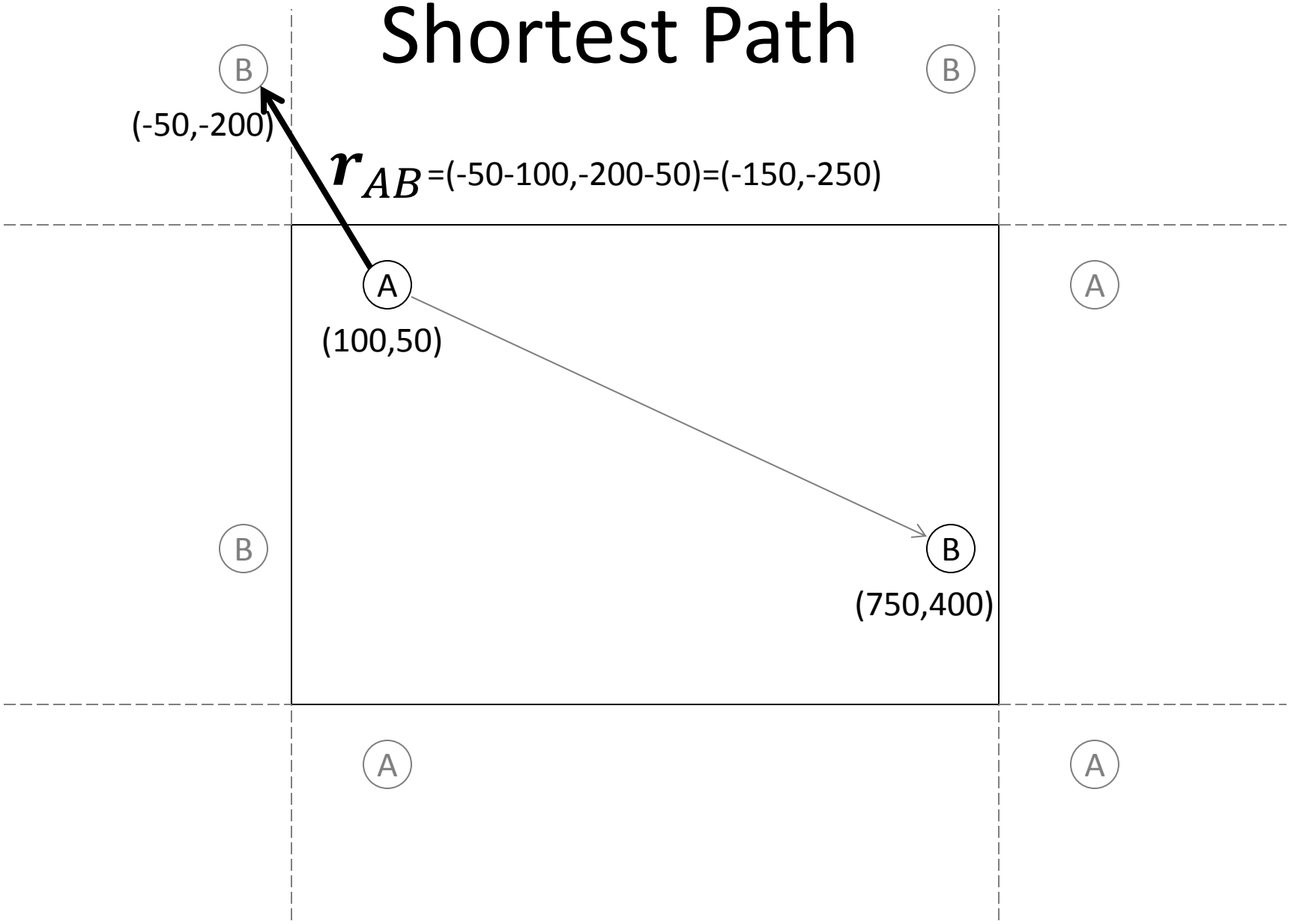
Vector Arithmetic



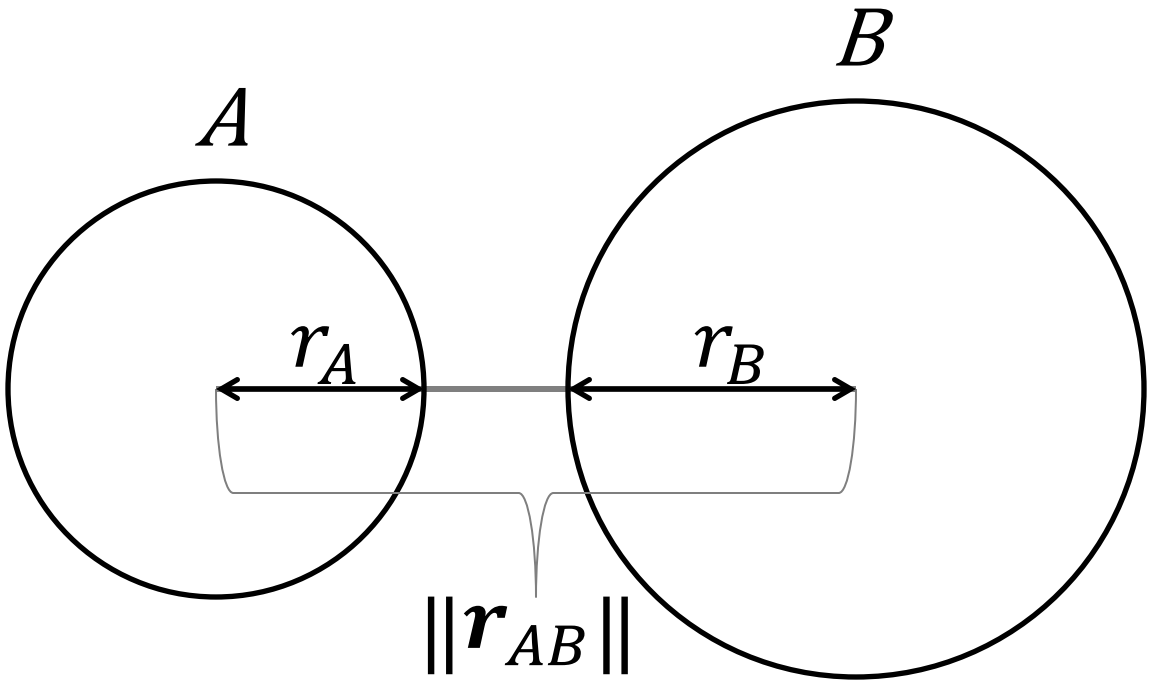
Wrapping


$\boldsymbol{p}^{(new)}$
(850,-200)
$\boldsymbol{p}^{(old)}$
(600,50)
$\boldsymbol{p}^{(new)}$
(50,400)
$\boldsymbol{p}^{(old)}$

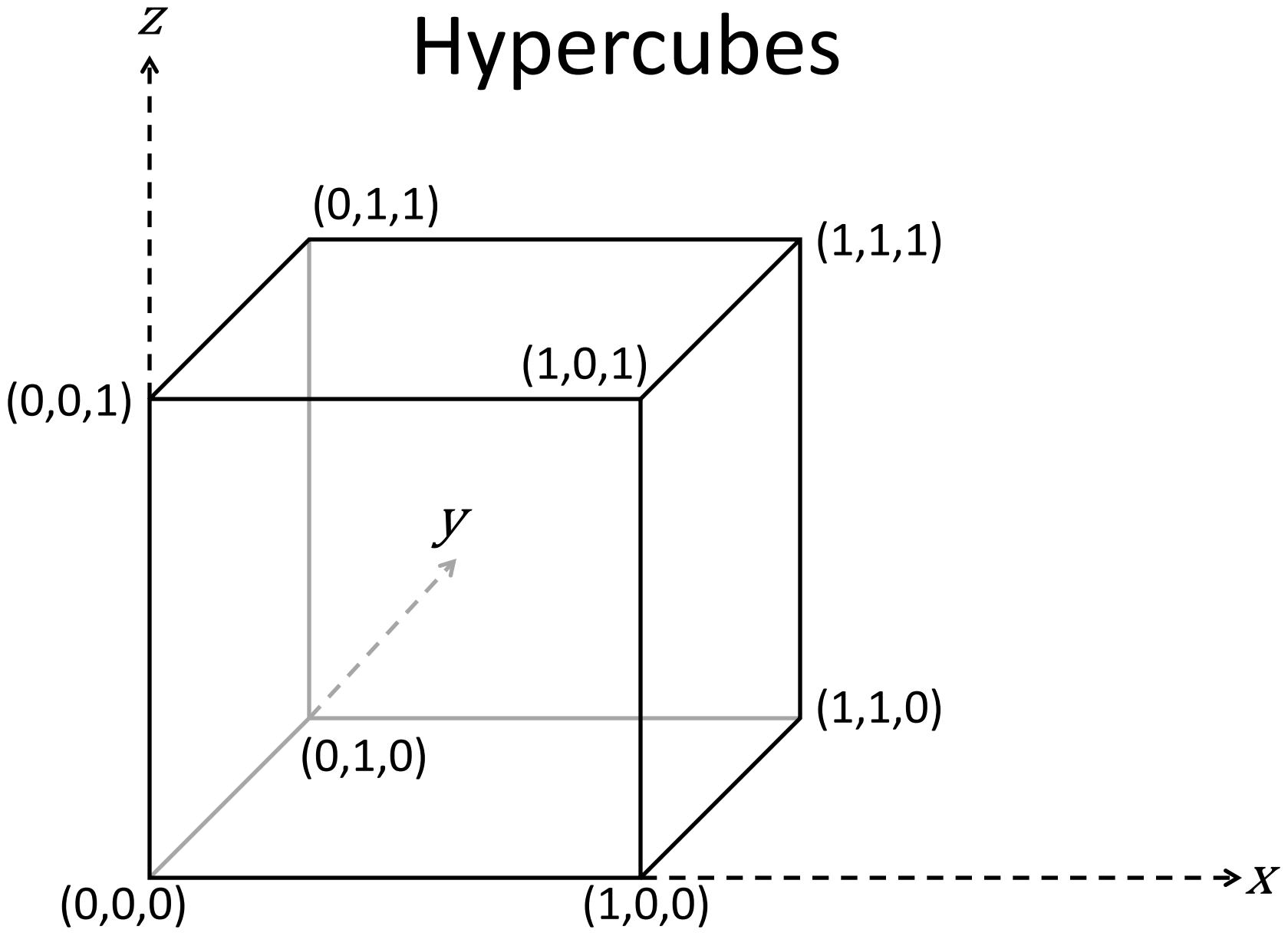
Shortest Path



Collision Detection



Hypercubes



Linear Algebra and Physics

- Matrix operations
  - Entry-wise operations
  - Matrix dot product
  - Matrix multiplication
  - Rotation Matrices
  - Round-off error with floating-point arithmetic
- Newtonian mechanics
  - Velocity, acceleration, Newton's laws
  - Gravitational force
  - Collision force

Inheritance and Design

- Issues with protected fields
  - http://programmers.stackexchange.com/questions/162643/why-is-clean-code-suggesting-avoiding-protected-variables
  - Reduces encapsulation: sub-classes can expose internal state
  - YAGNI: “You aren’t gonna need it”
  - LSP: Liskov Substitution Principle
  - OCP: Open(for extension)/Closed(for modification) principle

Inheritance and Design

- Extension vs. Composition
  - Extension: extending functionality of a super-class
  - Composition: wrapping another class in a field
  - BigInt example, ShapeList revisited
- Style: code reuse with super
  - Copy constructors
  - Standard Equals
  - Super examples

Reflectance

- The ability of your source code to “hold up a mirror” and inspect itself.
- Allows things like:
  - Manipulating fields (including private ones)
  - Converting between source code and Strings
    - “BasicSoldier”
    - “ShinyCoin”
- Requires certain execution environments
  - Not available in Applets (used on the internet)

The `Class` class

- Represents a data-type
  - Reference types: objects, enums
  - Primitive types
- Is generic!
- All constructors are *private.* Methods are used for acquiring a Class object:
  - myObj.getClass();
  - myPrimitive.class;
  - Class.forName("…");
  - And more: Reflectance example

The Reflection API

- Acquiring Class objects is just the tip of the ice-berg of a large API:
  - myClass.newInstance(); //instantiates the class
  - myClass.getMethods(); //enumerates the methods
  - There is a Method class
    - myMethod.invoke(…);
  - There is a Type class for generic type parameters
  - There is an Annotation class
    - @override
    - @Test