Domain to Design classes

- Design class should (almost always)
  - Represent domain classes
  - Refine domain classes
  - Add control, creator, and expert classes

Example

Which did you have?
- Ball
- Sphere
- Triangle
- Polygon
- Thing
- Shape
- Moving shape
- Moving object
- Velocity
- Walls
- Floor
- World
- Force
- Gravity
- Collision
- Position
- Item

Ball vs. Sphere vs. Shape

What is the relationship?

design heuristic

think like an object

Ball vs. Sphere vs. Shape

What is the relationship?
**Ball vs. Sphere vs. Shape**

What is the relationship?

```
Ball --> Shape
|       |
|       |
| Sphere|
```

**Shape vs. Moving**

What is the relationship?

```
Shape

- velocity

shape doesn't move? set velocity to 0
```

**Shape vs. Moving**

What is the relationship?

```
Shape

- Static shape
- Moving shape
```

**Shape vs. Moving**

```
Shape

- Static shape
- Moving shape

- sphere
- triangle
  ```
Shape vs. Moving

What is the relationship?

Inheritance vs. Composition

Design heuristics

- Think like an object
- Favor composition over inheritance
- Low coupling & high cohesion

Design heuristic

favor composition over inheritance

cohesion & coupling

- cohesion: how closely the operations in a routine/class are related
- coupling: the strength of a connection between two routines/classes
Design heuristics

- Think like an object
- Favor composition over inheritance
- Low coupling & high cohesion
- Only talk to your immediate friends (LoD-DPIC)

LoD-DPIC

- Law of Demeter: only talk to your immediate friends
- An object should only invoke methods of:
  - objects that are declared within it
  - objects that are parameters of the method
  - itself
  - objects that it creates

Liskov Substitution Principle (LSP)

As popularly stated:
A member of a derived class must also make sense when used as a member of the base class.
For example, if a method has an object of a class as an argument, the same method should be able to work with an object of a derived class.

As originally stated:
Let $\phi(x)$ be a property provable about objects $x$ of type $T$. Then $\phi(y)$ should be true for objects $y$ of type $S$, where $S$ is a subtype of $T$.

More design heuristics

- Objects as organism: Class objects should be responsible for their own behavior
- Use utility classes freely in place of primitives.
- Keep your classes light.
- Favor pointer members over instance members.
- Avoid forgery: do not keep the same data in more than one place.

Triangle/Ball world revisited

- evaluate your design relative to each design heuristic
- red-design