CSSE 220

Collision Handling without **instanceof**

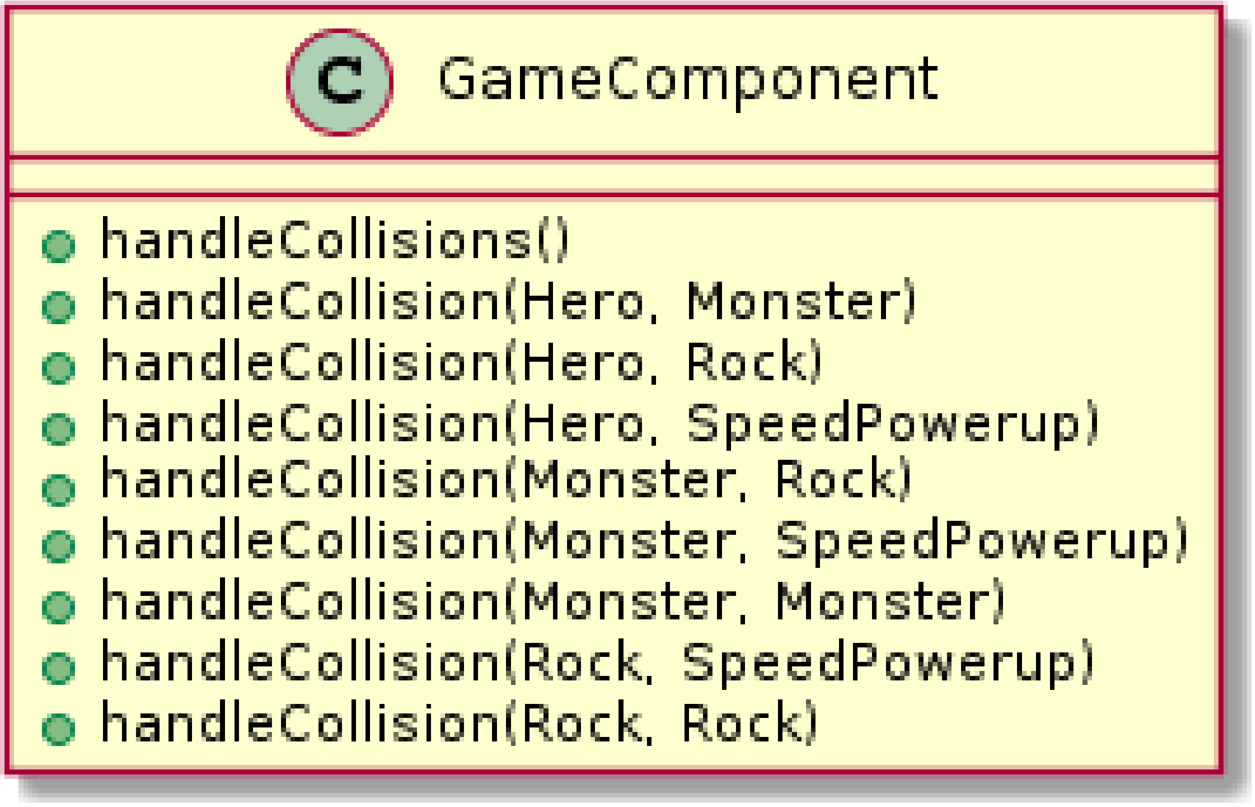
Checkout *DoubleDispatch* project from SVN

The problem

- Monsters can collide with rocks.
- Rocks can crush monsters.
- Players can collide with monsters.
- Players can be crushed by rocks.
- Players can take powerups.

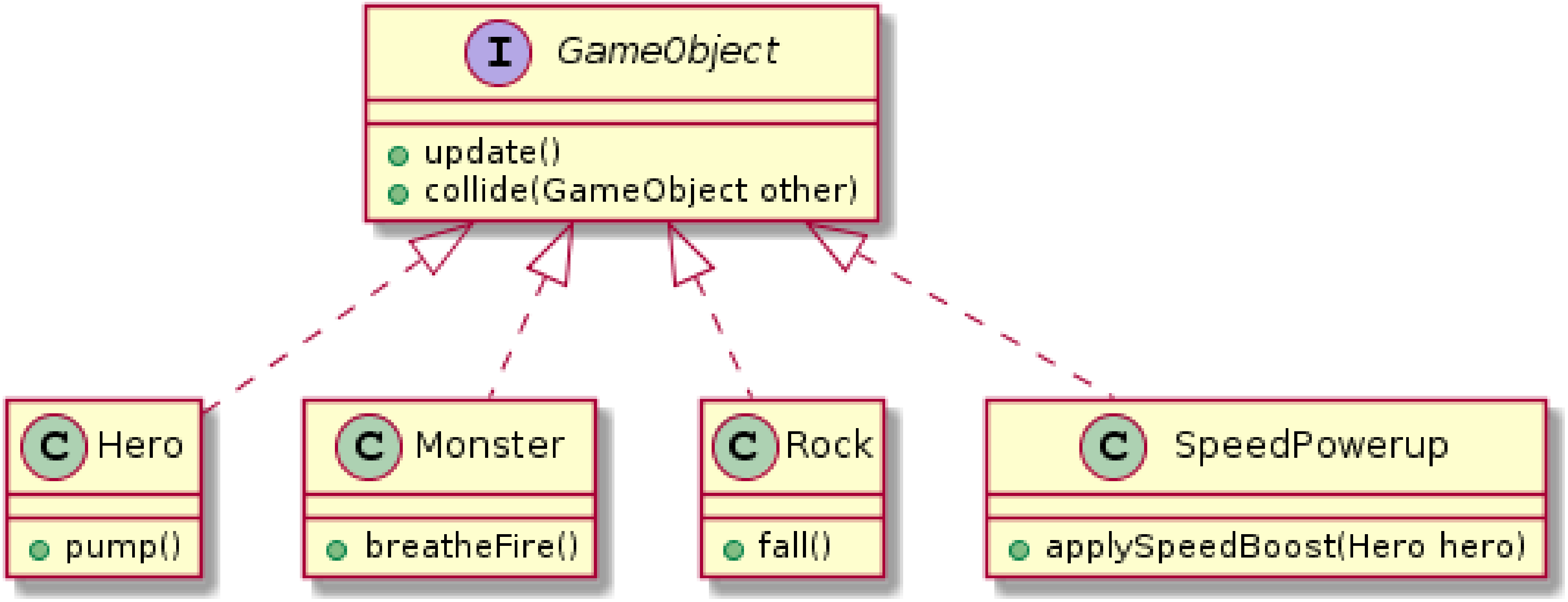
So many collisions! How do we handle them all?

What not to do


C
GameComponent
handleCollisions()
handleCollision(Hero, Monster)
handleCollision(Hero, Rock)
handleCollision(Hero, SpeedPowerup)
handleCollision(Monster, Rock)
handleCollision(Monster, SpeedPowerup)
handleCollision(Monster, Monster)
handleCollision(Rock, SpeedPowerup)
handleCollision(Rock, Rock)

Why is this design bad?

Slightly better?



But tempts you to use **instanceof**…

A bad Player.collide(GameObject o1)

```
// player has landed on o1
if(o1 instanceof SpeedPowerUp) {
   // code to increase speed
}
if(o1 instanceof LifePowerUp) {
 // code to increase life
}
```

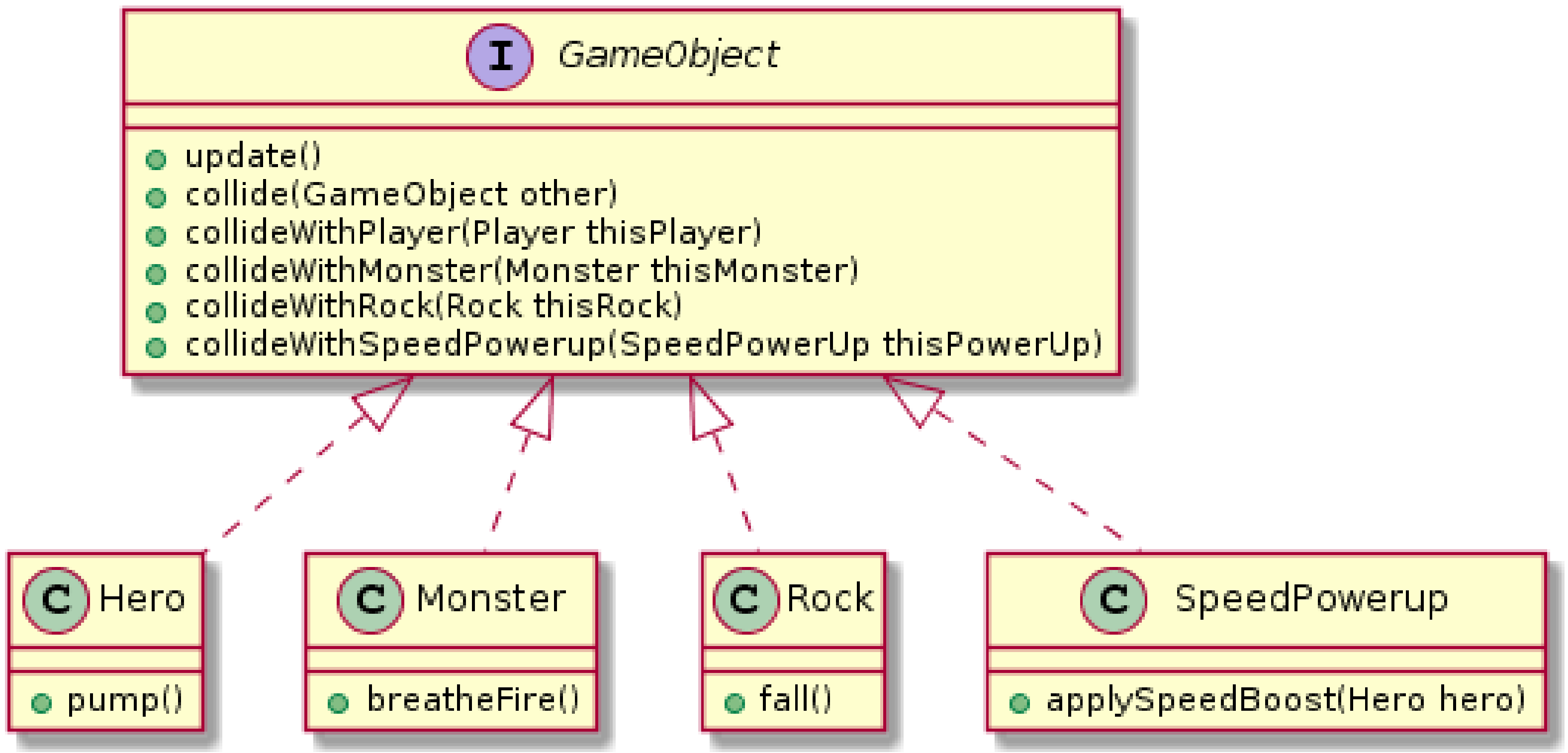
<u>Same</u> Bad Idea

```
//player has landed on o1
if(o1.type().equals(“SpeedPowerUp”))
{
   //code to increase speed
}
if(o1.type().equals(“LifePowerUp”)) {
   //code to increase life
}
```

instanceof - in general

- **instanceof** is like static. It is dead to you.
- Instead: add **new interface methods**.
- Recall this is called **polymorphism**.

Polymorphic Solution


I GameObject
update()
collide(GameObject other)
collideWithPlayer(Player thisPlayer)
collideWithMonster(Monster thisMonster)
collideWithRock(Rock thisRock)
collideWithSpeedPowerup(SpeedPowerUp thisPowerUp)
C Hero
pump()
C Monster
breatheFire()
C Rock
fall()
C SpeedPowerup
applySpeedBoost(Hero hero)

Polymorphic Solution

```
o1.collideWithPlayer(player);

// in SpeedPowerUpClass
void collideWithPlayer(Player p) {
   // code to increase speed
}
```

What made this work

- We knew one of the objects was the Player.
- In general:
  - Objects know **their own type**.
  - They also know the **other object's interface**.

```
o1.collideWithPlayer(player);

// in SpeedPowerUpClass
void collideWithPlayer(Player p) {
   // code to increase speed
}
```

Double Dispatch

Objects make collide______() calls on each other until one decides to handle the collision.

```
public class SpeedPowerUp extends GameObject {
        public void collide(GameObject other) {
                other.collideWithSpeedPowerUp(this);
        }
```

The other object decides for itself how to respond.

```
        public void collideWithPlayer(Player thisPlayer) {
                //do specific action to player
                thisPlayer.speedUp();
        }
```

The Player called speedPowerUp.collideWithPlayer(thisPlayer)

```
        …
}
```

See DoubleDispatch in repo

Work time

Be sure everyone is getting a chance to drive.

TEAM PROJECT