Classes & Objects

Class

[class]

Dog
size breed name
bark()

[**class name**]

[**instance variables**]
knows / state

[**methods**]
does / behavior

The *size* of a dog is big.

A dog *barks*.

Class

[class]

[class name]

[instance variables]
knows / state

static variable

[methods]
does / behavior

The *size* of a dog is big.

A dog *barks*.

Class

[class]

Dog
size breed name color of blood
bark()

[**class name**]

[**instance variables**]
knows / state

[**methods**]
does / behavior

Dog : class
a dog : object (instance of Dog)

The *size* of Dog is big.

Dog *barks*.

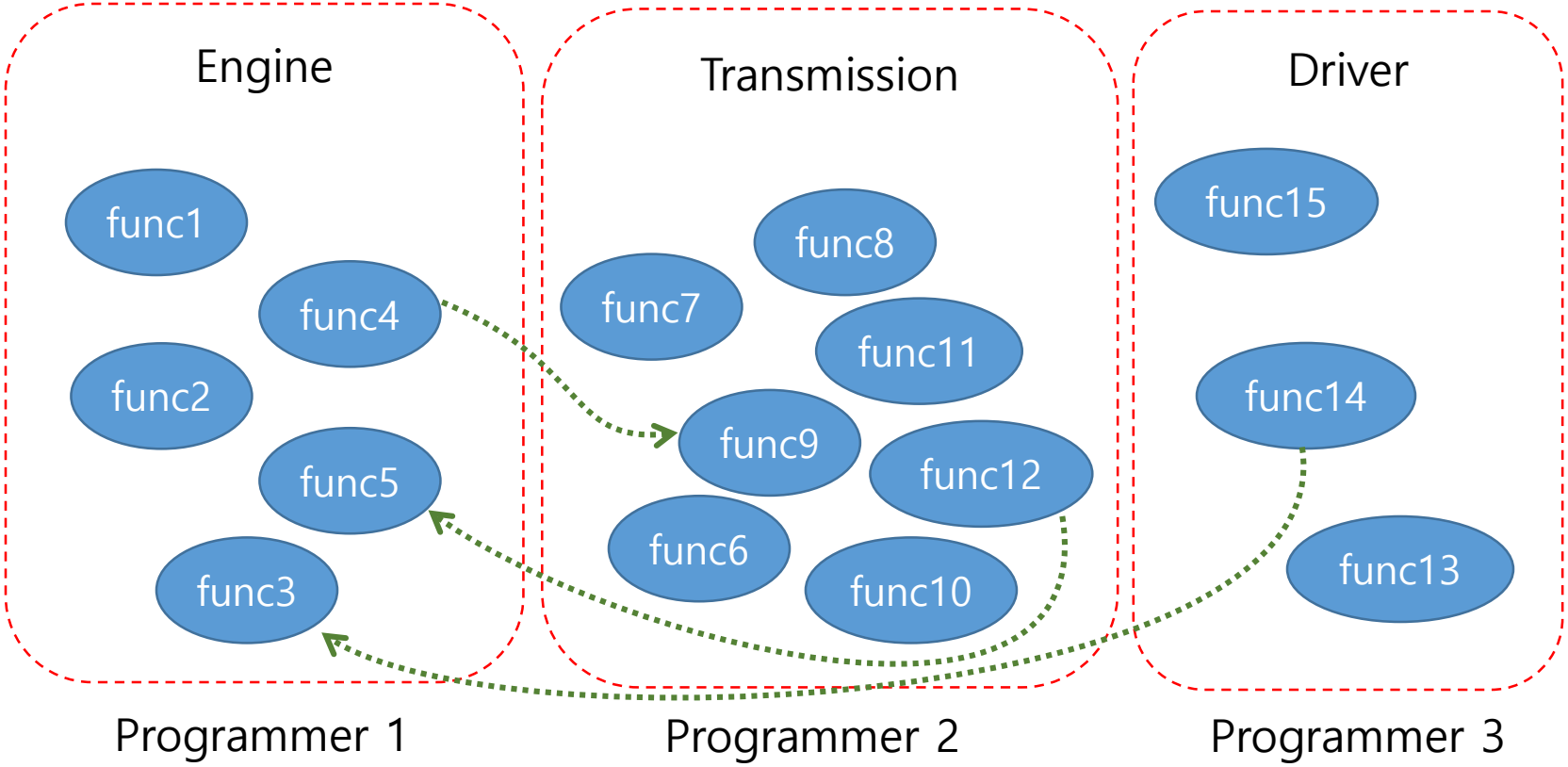
Class

[class]


```
class Dog {
   int size;                                instance variables
   String breed;
   String name;

                                            static variable
   static Color color_of_blood = RED;

   void bark() {                            methods
      System.out.println("Ruff!");
   }
}

public class DogsPlayground {

                                            main method for testing
   public static void main(String [] args) {
       Dog marianne = new Dog();
       marianne.size = 40;
       Color bloodColor = Dog.color_of_blood;
   }

}
```

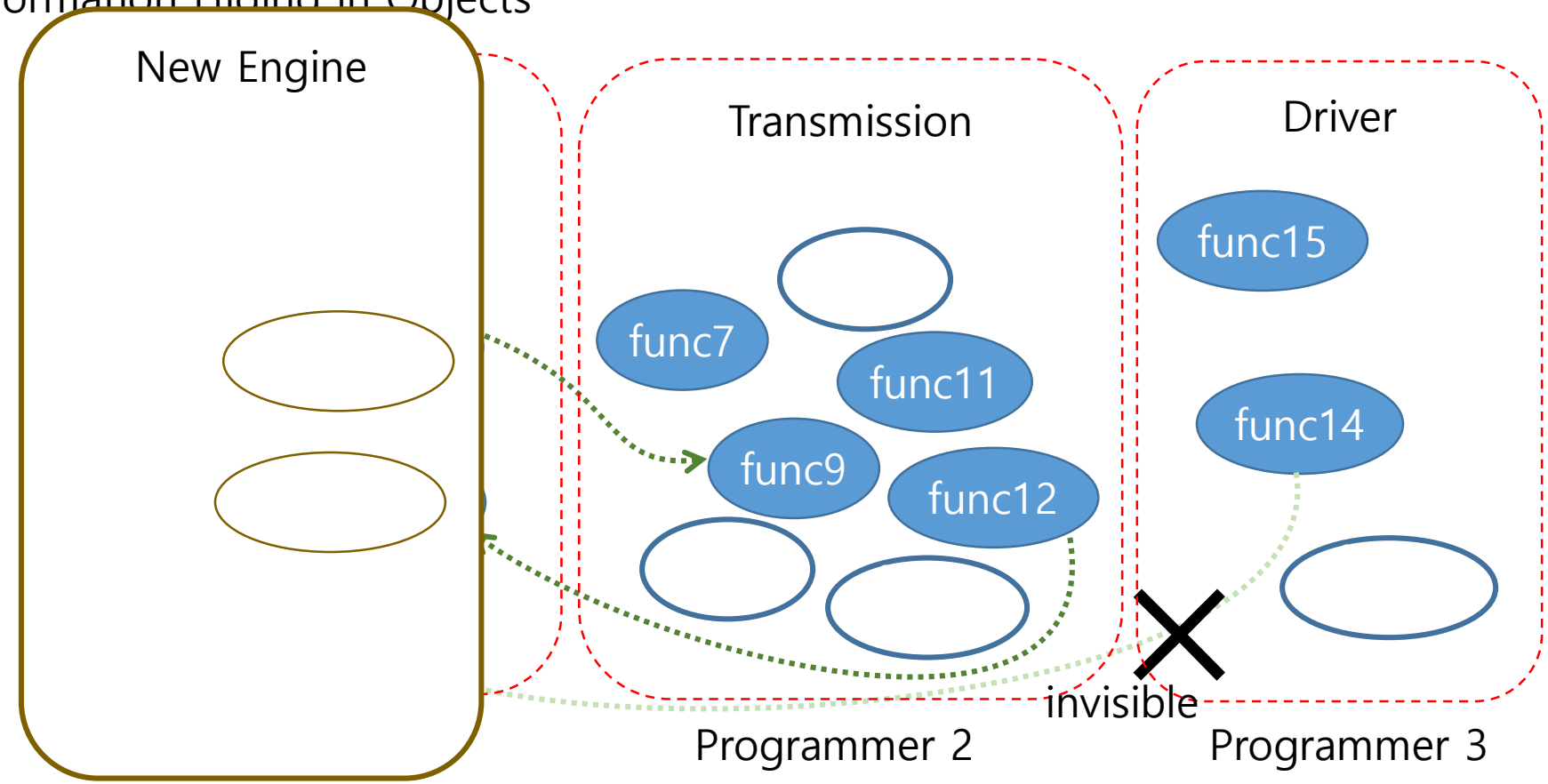
Benefits of building with objects

Modular Programming Development



Benefits of building with objects

Information Hiding in Objects



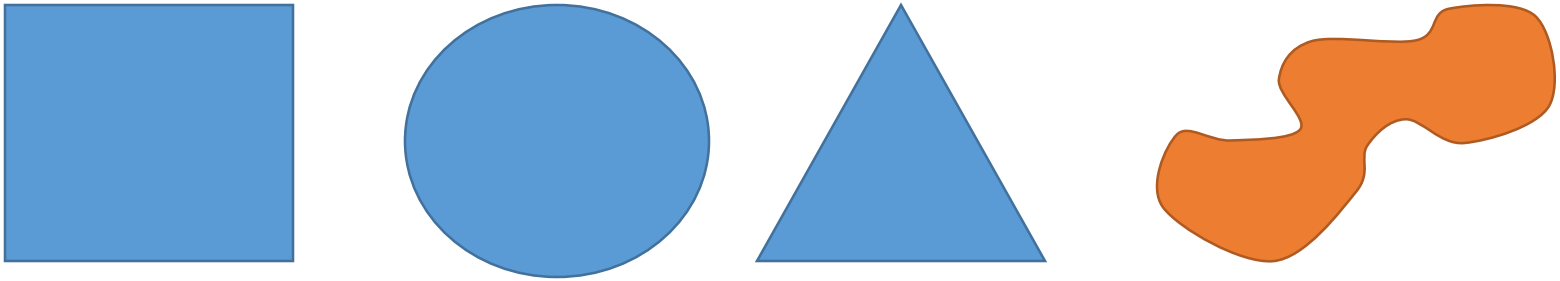
Benefits of building with objects

- **Information-hiding**
  - Implementation details remain hidden from outside
- **Modularity**
  - A class source code is written and maintained independently of those of others.
- **Code re-use**
  - Existing objects can be reused in other programs without modifications.
- **Plug-ability and easy debugging**
  - A problematic object can be easily replaced with other object, analogously to fixing mechanical problems.

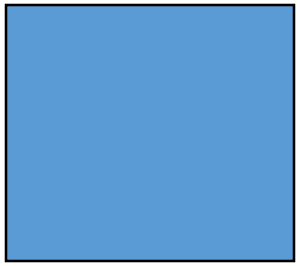
→ High Software Productivity & Maintainability

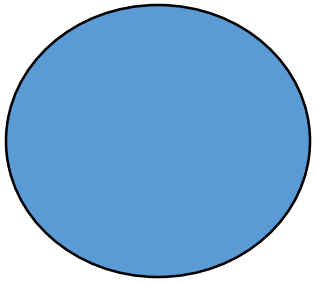
Chair Wars

- Competitors
  - Larry: procedural programmer
  - Brad: object-oriented programmer
- Reward
  - Aeron chair

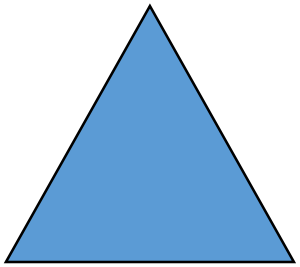


Initial Spec. v1.0





- Spec
  - rotate
  - playSound (.aif file)



- Progress
  - Larry (Procedural Programmer)
    - What are the things this program has to do?
    - What procedures do we need?
  - Brad (Object-Oriented Programmer)
    - What are the things in this program?
    - Who are the key players? – the shapes.

Procedural Programming (1)

- Procedures (functions in C)

```
enum SHAPE_TYPE {SQUARE, CIRCLE, TRIANGLE};

typedef struct {
    SHAPE_TYPE type;
} SHAPE;

typedef struct {
    SHAPE_TYPE type;
    int left, top, size;
} SQUARE;

typedef struct {
    SHAPE_TYPE type;
    int cx, cy, radius;
} CIRCLE;

typedef struct {
    SHAPE_TYPE type;
    int v1x, v1y, v2x, v2y, v3x, v3y;
} TRIANGLE
```

Procedural Programming (2)

- Procedures (functions in C)

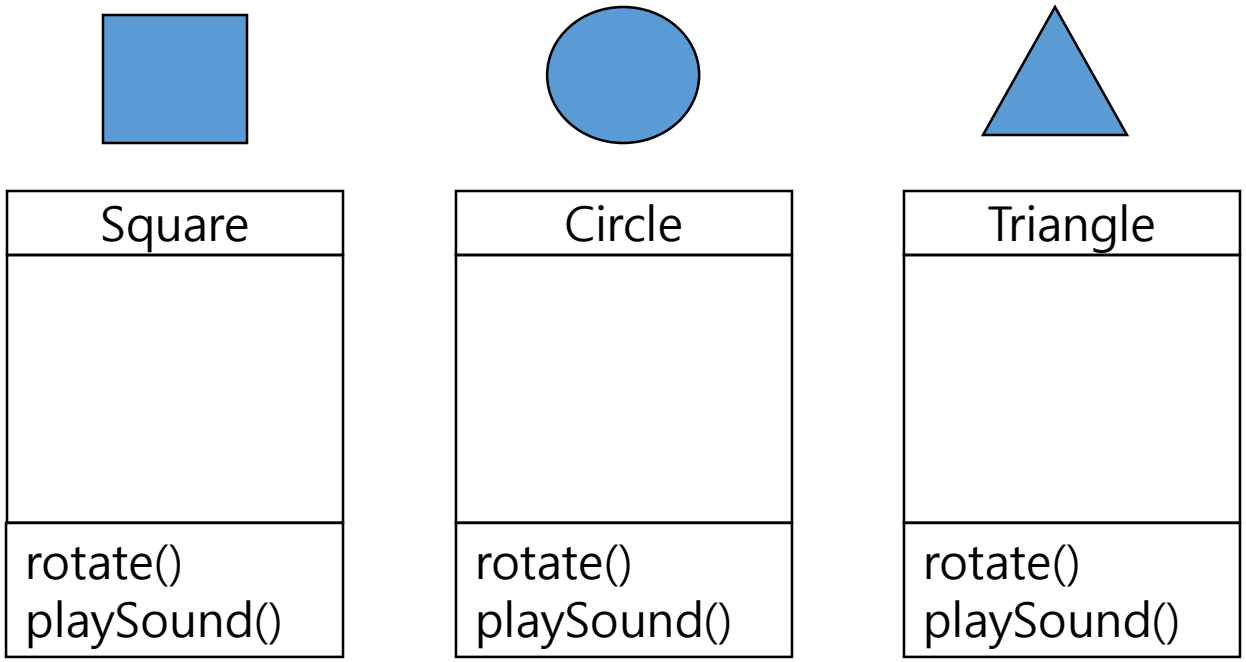
```
void rotate (SHAPE * shape) {
    switch (shape->type) {
    case SQUARE:
        {   SQUARE * square = (SQUARE *)shape;
            // rotate the square
        }
        break;
    case CIRCLE:
        // rotate the circle
        break;
    case TRIANGLE:
        // rotate the triangle
        break;
    default:
        // invalid arguments
    }
}

void playSound (void * shape) { ... }
```

Procedural Programming (3)

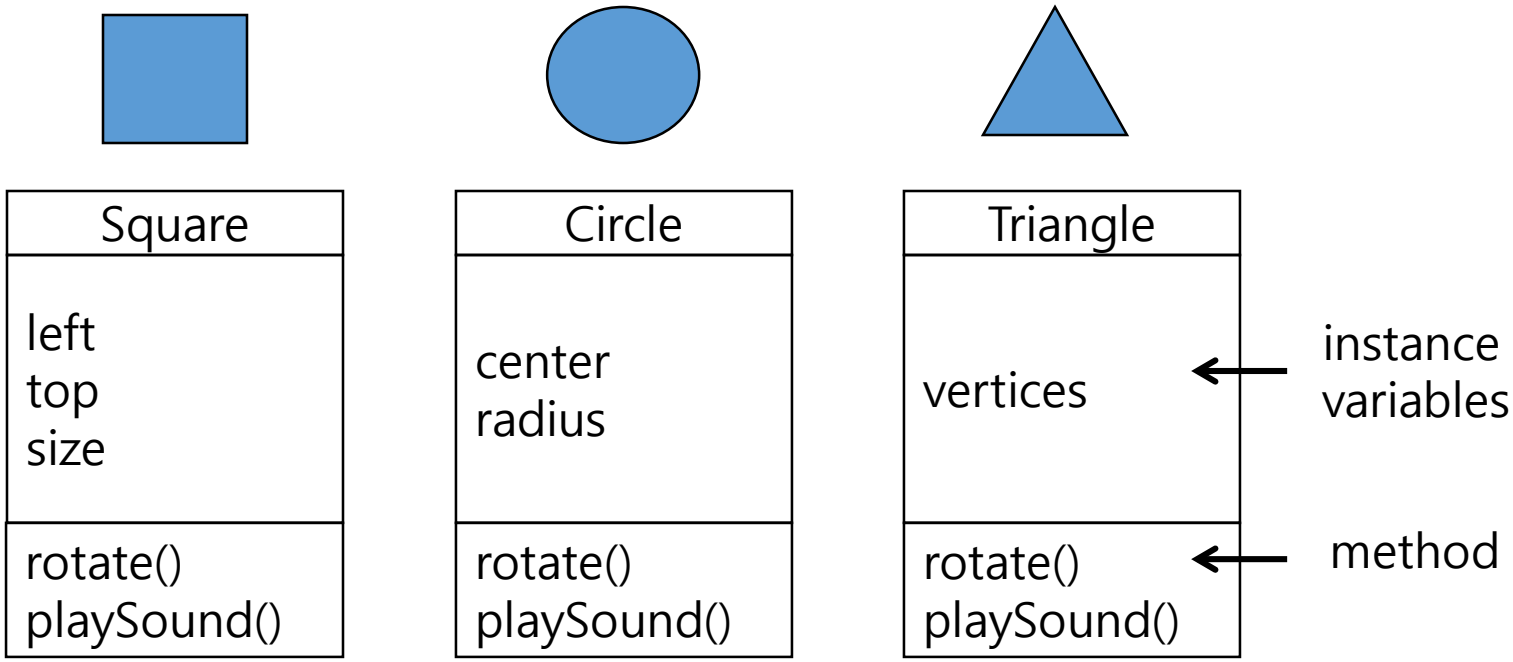
- Simply…

```
rotate (shape) {
    // make the shape rotate 360 degrees.
}

playSound (shape) {
    // use shape to lookup which
    // AIF sound to play, and play it
}
```

Object-Oriented Prog. (1)

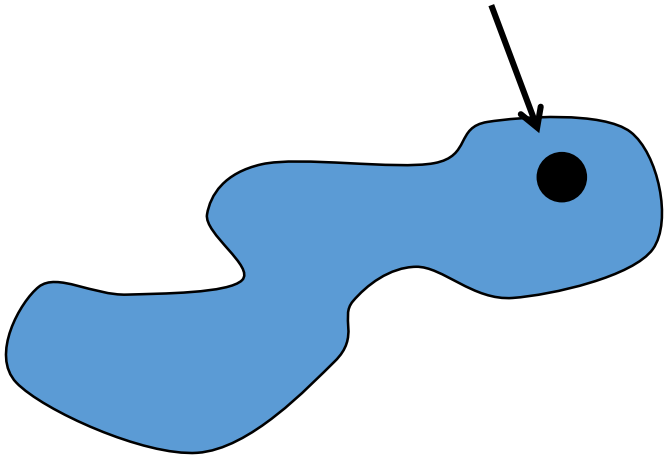


Object-Oriented Prog. (2)



Revised Spec. v1.1

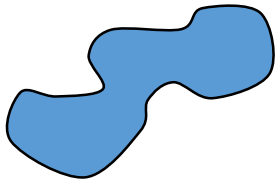
- Changed spec
  - An amoeba shape
  - Rotate using a given position (x, y)
  - Play .hif file for an amoeba
- Progress
  - Larry (procedural programmer)
    - Should change rotate() and playSound().
  - Brad (object-oriented Programmer)
    - Write a new class.



Procedural Programming (4)

- Procedures (functions in C)

```
enum SHAPE_TYPE {SQUARE, CIRCLE, TRIANGLE, AMOEBA};

typedef struct {
    SHAPE_TYPE type;
} SHAPE;

...

typedef struct {
    SHAPE_TYPE type;
    int num_of_control_points;
    POINT * control_points;
} AMOEBA;
```



Procedural Programming (5)

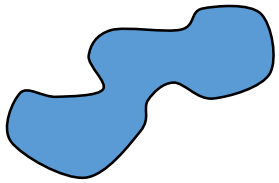
- Procedures (functions in C)

```
void rotate (SHAPE * shape, POINT center) {
    switch (shape->type) {
    case SQUARE:
        {   SQUARE * square = (SQUARE *)shape;
            // rotate the square
        }
        break;
    case CIRCLE:
        // rotate the circle
        break;
    case TRIANGLE:
        // rotate the triangle
        break;
    case AMOEBA:
        // rotate the amoeba using the variable center
        break;
    default:
        // invalid arguments
    }
}
```

Procedural Programming (6)

- Procedures (functions in C) (alternative)

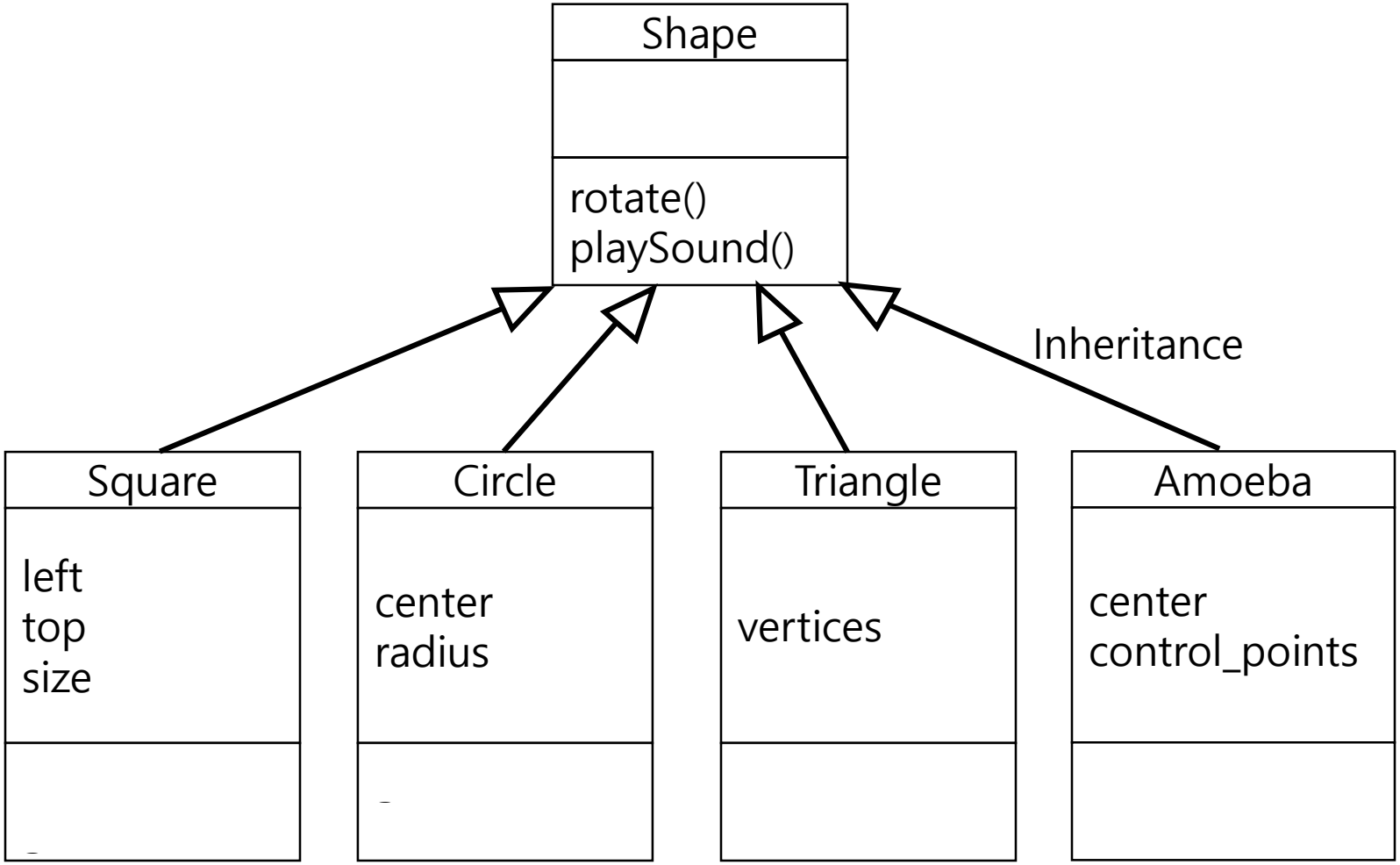
```
enum SHAPE_TYPE {SQUARE, CIRCLE, TRIANGLE, AMOEBA};

typedef struct {
    SHAPE_TYPE type;
} SHAPE;

...

typedef struct {
    SHAPE_TYPE type;
    int num_of_control_points;
    POINT * control_points;
    POINT rotation_center;
} AMOEBA
```



Object-Oriented Prog. (3)

Square	Circle	Triangle	Amoeba
left top size	center radius	vertices	center control_points
rotate() playSound()	rotate() playSound()	rotate() playSound()	rotate() playSound()

Object-Oriented Prog. (4)



Making & Testing Movie Objs.

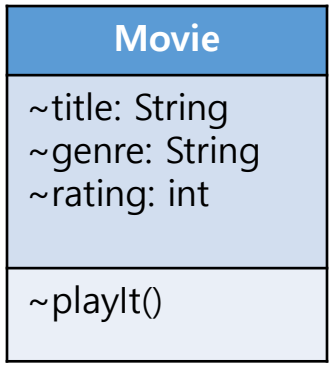

```
class Movie {
   String title;
   String genre;
   int rating;

   void playIt() {
      System.out.println("Playing the movie");
   }
}
```

```
public class MovieTestDrive {
   public static void main(String[] args) {
      Movie one = new Movie();
      one.title = "Gone with the Stock";
      one.genre = "Tragic";
      one.rating = -2;
      Movie two = new Movie();
      // ...
   }
}
```

Class Diagram (1)

- From Donald Bell, "UML basics: The class diagram", 2004.


```
class Movie {
   String title;
   String genre;
   int rating;

   void playIt() {
      System.out.println("Playing the movie");
   }
}
```



[instance variable]
name : attribute type = default value

[method]
name(parameter list) : type of value returned

UML Visibility Mark

For instance variables and methods

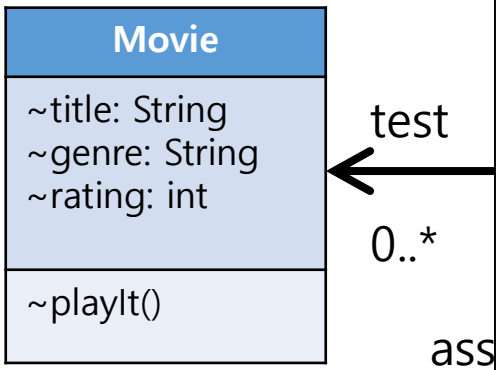
Mark	Visibility type	Java access modifier
+	public	public
-	private	private
#	protected	protected
~	package(-private)	*no modifier, default;*

Java access modifiers

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Class Diagram (2)

- Relationship




```
public class MovieTestDriver {
   public static void main(String[] args) {
      Movie one = new Movie();
      one.title = “Gone with the Stock”;
      one.genre = “Tragic”;
      one.rating = -2;
      Movie two = new Movie();
      two.title = “Lost in Cubicle Space”;
      two.genre = “Comedy”;
      two.rating = 5;
      two.playIt();
      Movie three = new Movie();
      three.title = “Byte Club”;
      three.genre = “Tragic but ultimately uplifting”;
      three.rating = 127;
   }
}
```

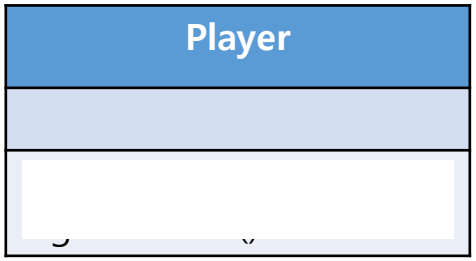
Guessing Game

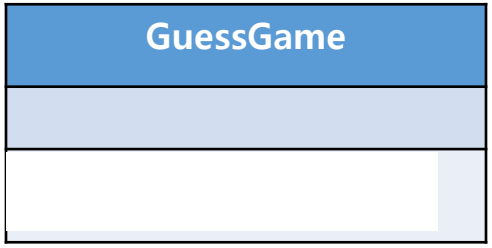
- What objects are playing in the game?
  - Game Launcher
  - Game Players (two players)
    1. One who picks a number and asks the other to guess.
       → Computer
    2. One who guesses a number of the opponent.
       → Human player

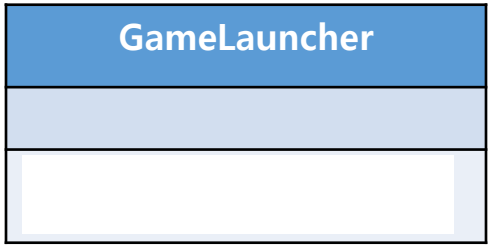
How many classes are needed ?

3 classes: **GameLauncher**, **GameGuess**, **Player**

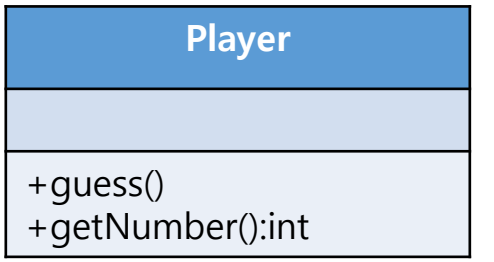
Guessing Game (1)

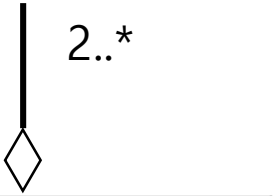


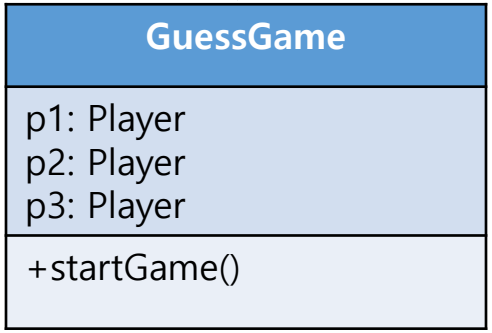


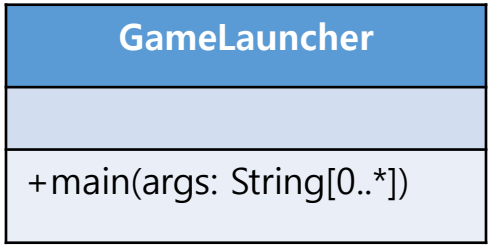


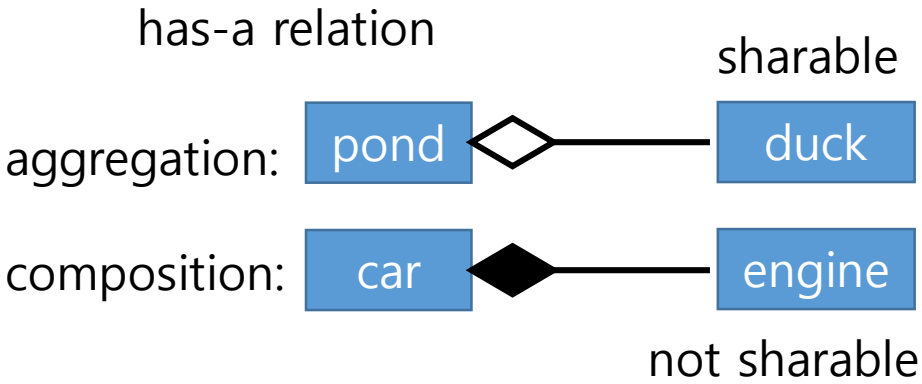
Guessing Game (1)



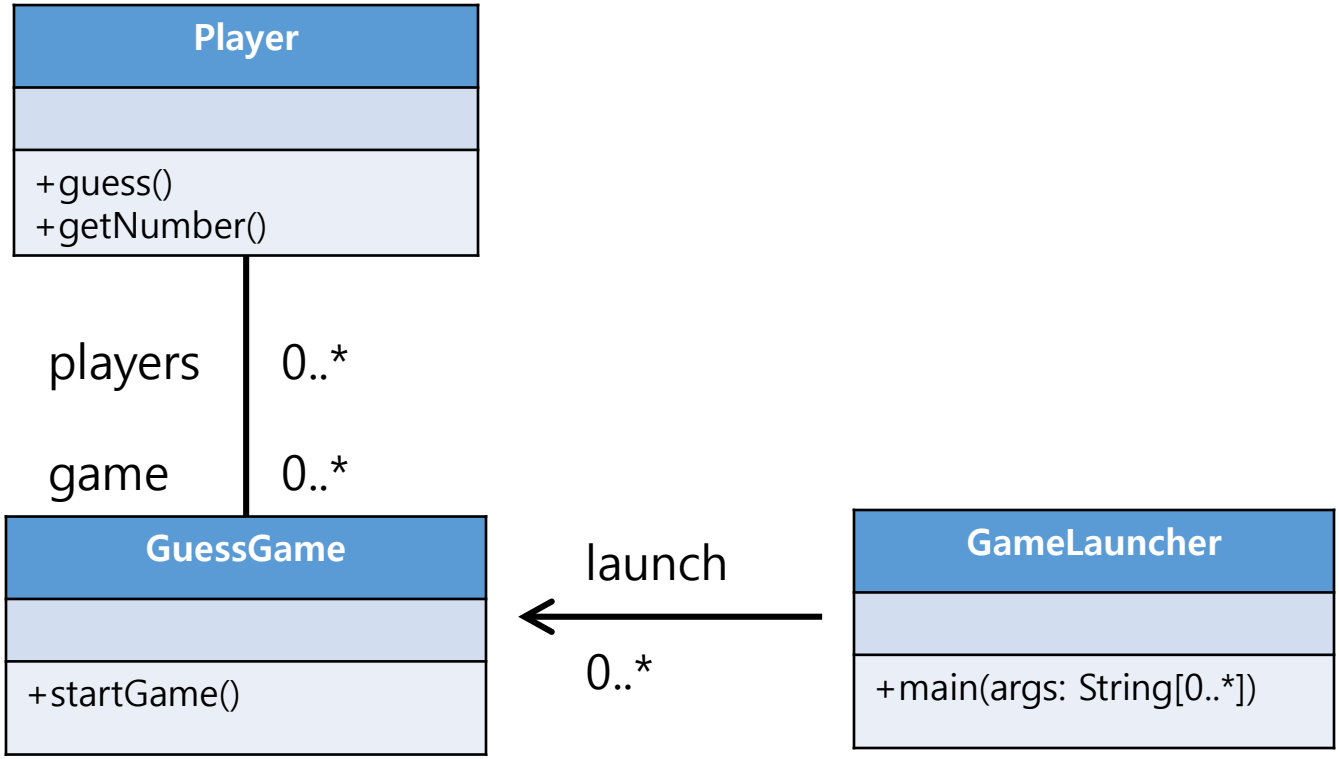








Guessing Game (1)



Guessing Game (2)

```
public class GuessGame{
   Player p1;
   Player p2;
   Player p3;

   public void startGame() {
      p1 = new Player();
      p2 = new Player();
      p3 = new Player();
      int guessp1 = 0;
      int guessp2 = 0;
      int guessp3 = 0;
      boolean p1isRight = false;
      boolean p2isRight = false;
      boolean p3isRight = false;
      int targetNumber = (int) (Math.random() * 10);
      System.out.println("I'm thinking of a number between 0 and 9...");
      while(true) {
         System.out.println("Number to guess is " + targetNumber);
         p1.guess();
         p2.guess();
         p3.guess();
         guessp1 = p1.getNumber();
         System.out.println("Player one guessed " + guessp1);
         guessp2 = p2.getNumber();
```

```
p3 = new Player();
int guessp1 = 0;
int guessp2 = 0;
int guessp3 = 0;
boolean p1isRight = false;
boolean p2isRight = false;
boolean p3isRight = false;
int targetNumber = (int) (Math.random() * 10);
System.out.println("I'm thinking of a number between 0 and 9...");
while(true) {
   System.out.println("Number to guess is " + targetNumber);
    p1.guess();
    p2.guess();
    p3.guess();
    guessp1 = p1.getNumber();
    System.out.println("Player one guessed " + guessp1);
    guessp2 = p2.getNumber();
    System.out.println("Player one guessed " + guessp2);
    guessp3 = p3.getNumber();
    System.out.println("Player one guessed " + guessp3);
    if (guessp1 == targetNumber) {
       p1isRight = true;
    }
   if (guessp2 == targetNumber) {
       p2isRight = true;
    }
    if (guessp3 == targetNumber) {
       p3isRight = true;
    }
    if (p1isRight || p2isRight || p3isRight) {
       System.out.println("We have a winner!");
       System.out.println("Player one got it right? " + p1isRight);
       System.out.println("Player two got it right? " + p2isRight);
```

```
        System.out.println("Number to guess is " + targetNumber);
        p1.guess();
        p2.guess();
        p3.guess();
        guessp1 = p1.getNumber();
        System.out.println("Player one guessed " + guessp1);
        guessp2 = p2.getNumber();
        System.out.println("Player one guessed " + guessp2);
        guessp3 = p3.getNumber();
        System.out.println("Player one guessed " + guessp3);
        if (guessp1 == targetNumber) {
           p1isRight = true;
        }
        if (guessp2 == targetNumber) {
           p2isRight = true;
        }
        if (guessp3 == targetNumber) {
           p3isRight = true;
        }
        if (p1isRight || p2isRight || p3isRight) {
           System.out.println("We have a winner!");
           System.out.println("Player one got it right? " + p1isRight);
           System.out.println("Player two got it right? " + p2isRight);
           System.out.println("Player three got it right? " + p3isRight);
           break;
        } else {
           System.out.println("Players will have to try again.");
        }
     }
  }
}
```

Guessing Game (3)

```
public class Player {
   private int number = 0;
   public void guess() {
      number = (int) (Math.random() * 10);
      System.out.println(“I’m guessing “ + number);
   }
   public int getNumber() {
      return number;
   }
}

public class GameLauncher {
   public static void main (String[] args) {
      GuessGame game = new GuessGame();
      game.startGame();
   }
}
```

Term

- Class
- Object (instance)
- Instance variable (member variable, attribute)
- Method (member function)
- Inheritance

Q&A