# CS1013 – Programming Project

Dr. Gavin Doherty

ORI LG.19

Gavin.Doherty@cs.tcd.ie

Trinity College Dublin

DSG Distributed Systems Group

# Program structure

setup()

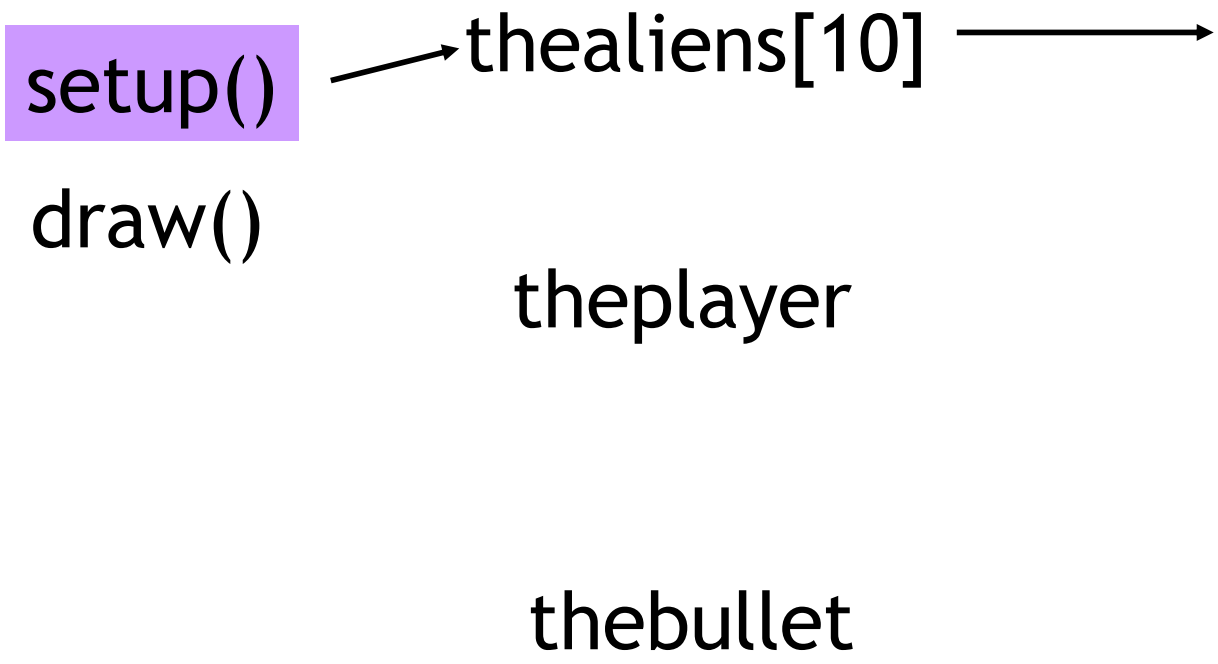
draw()

thealiens

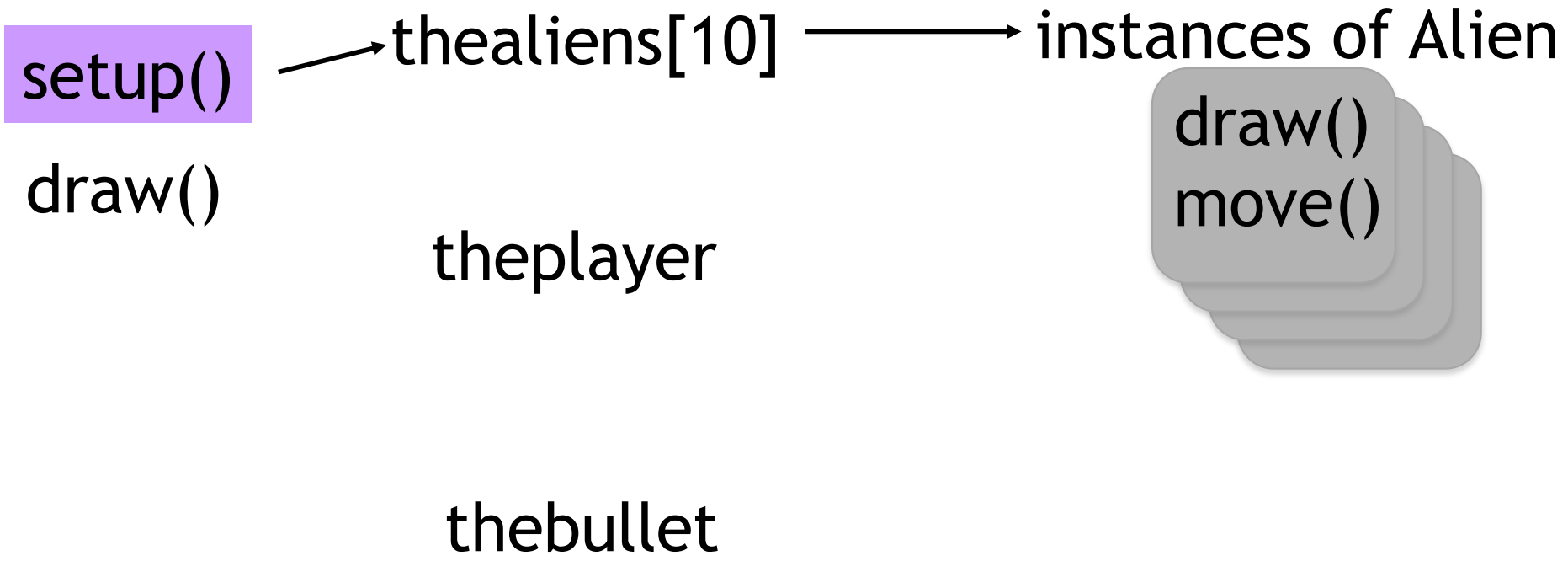
theplayer

thebullet

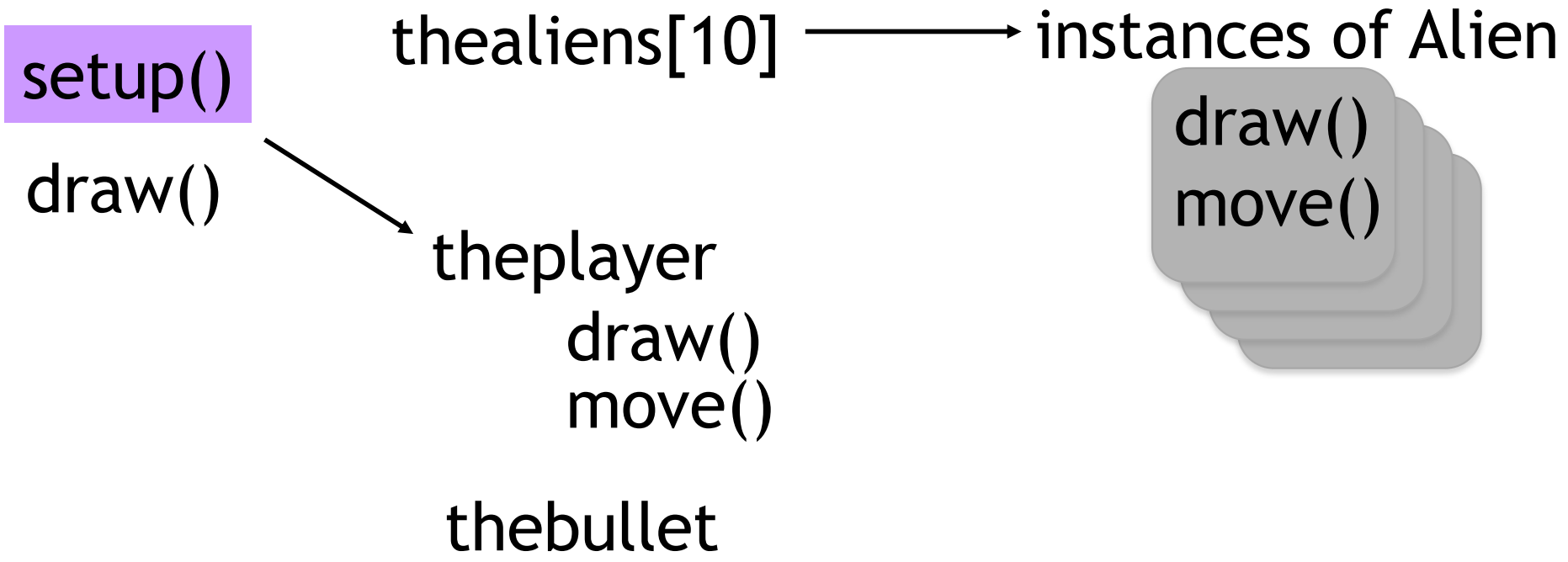
# Program structure



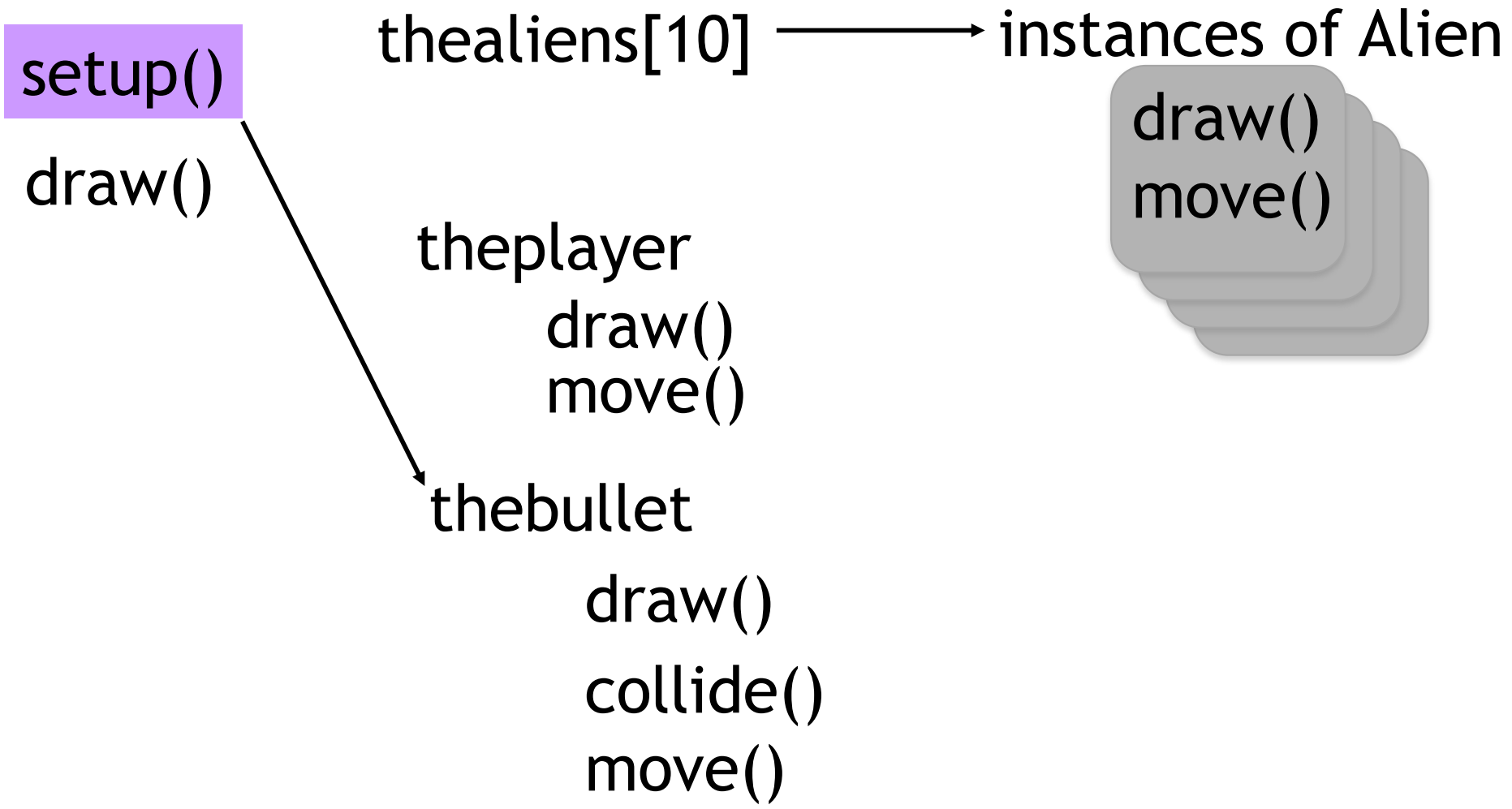
# Program structure



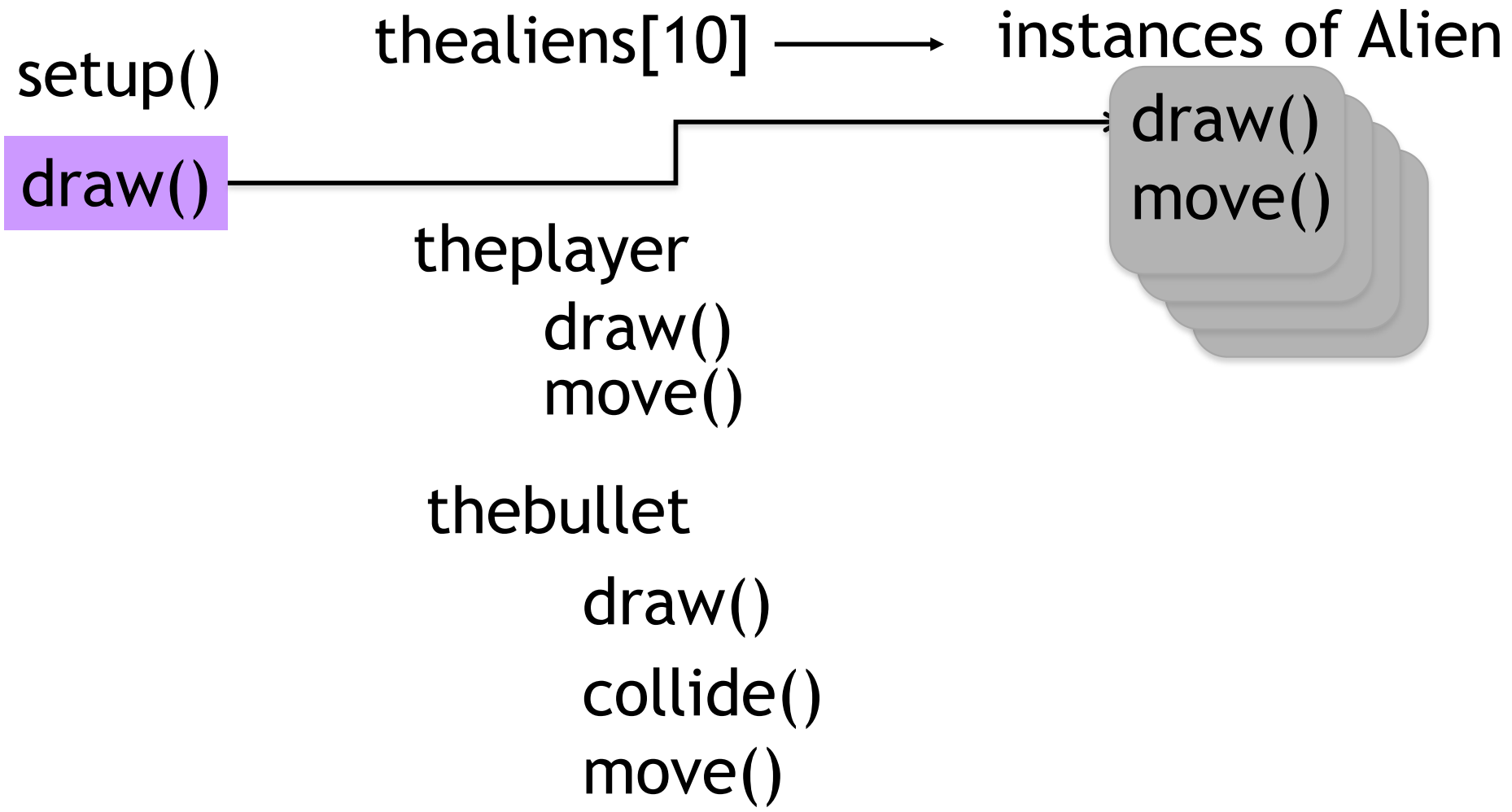
# Program structure



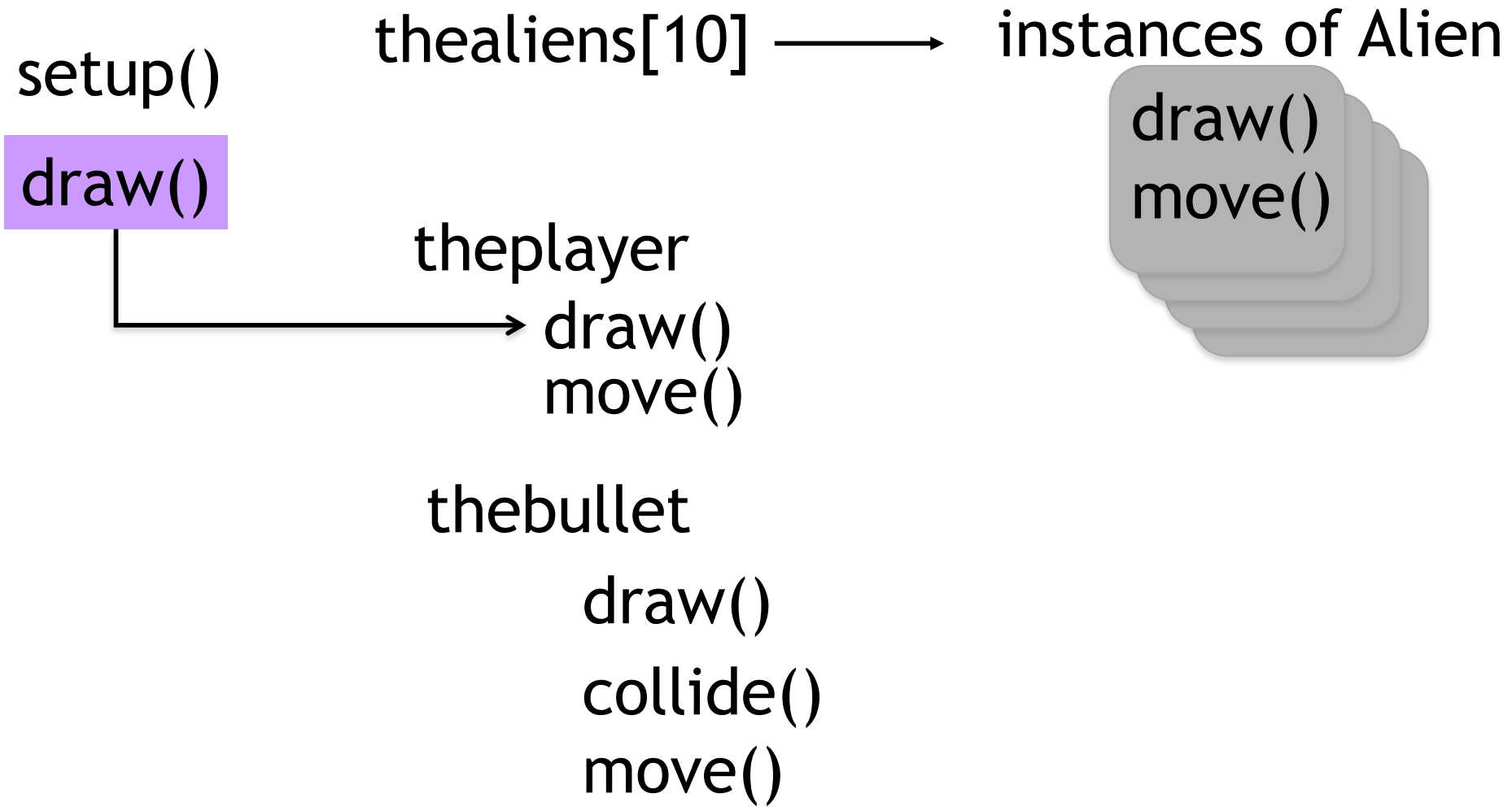
# Program structure



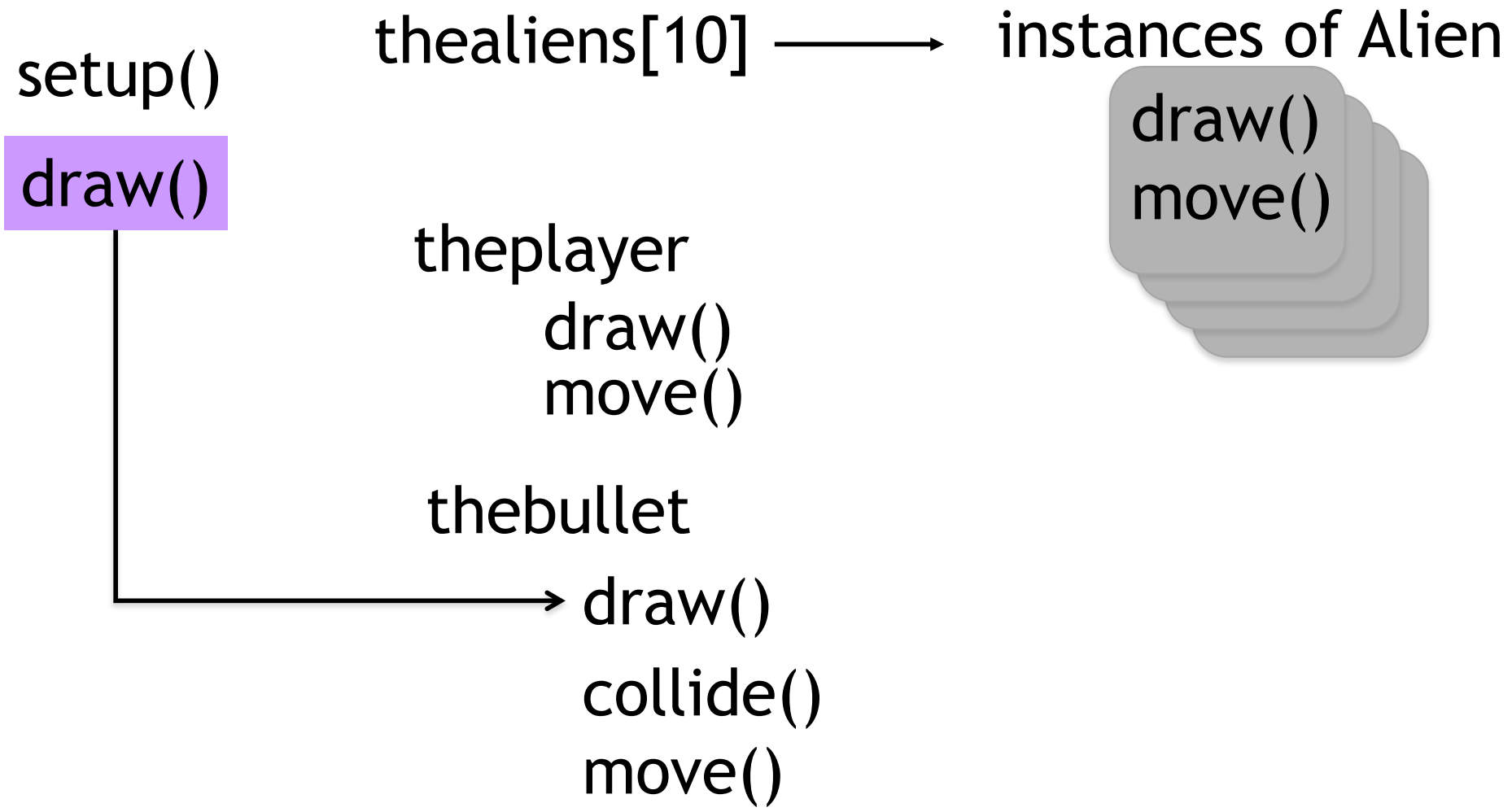
# Program structure



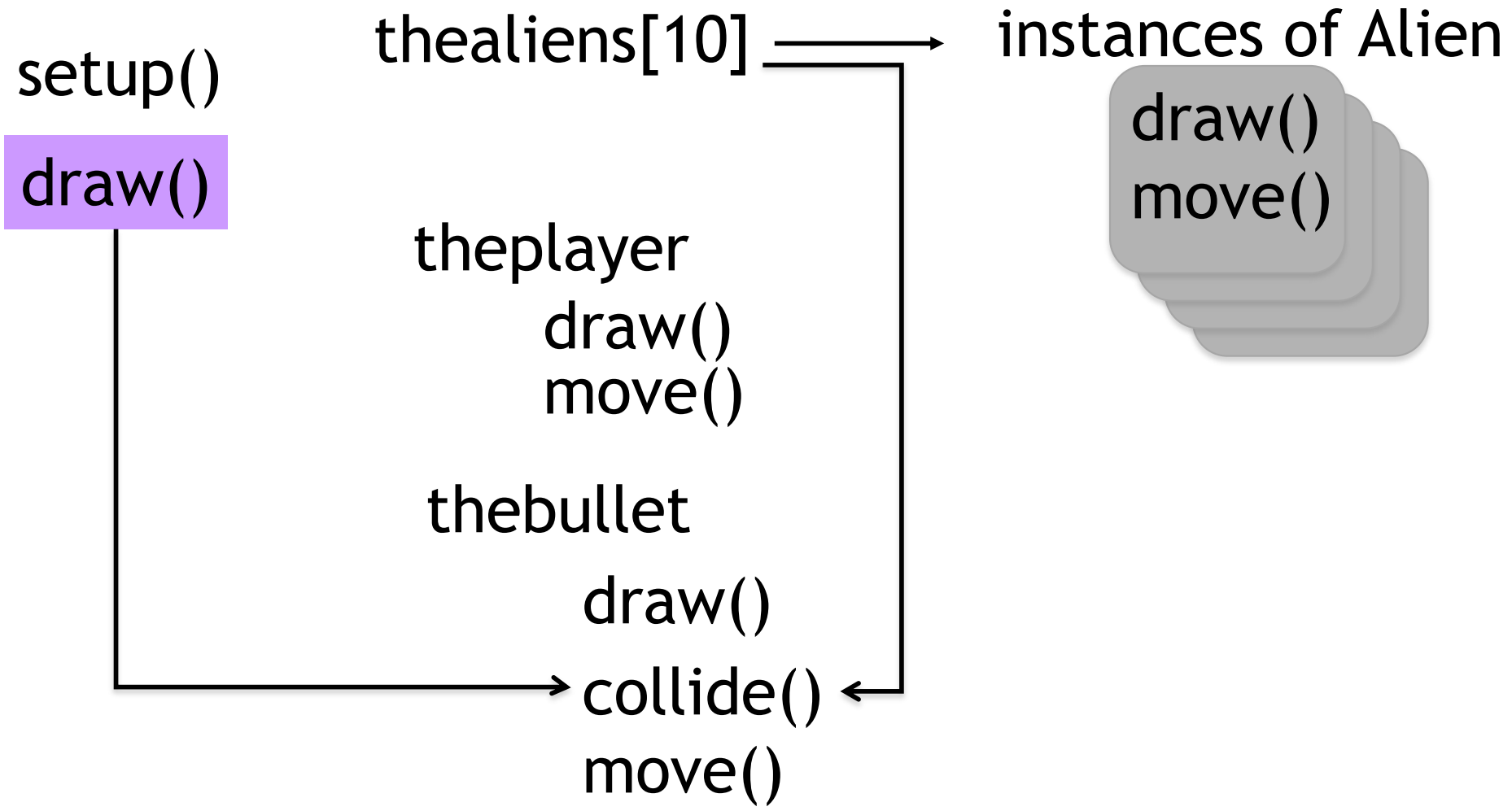
# Program structure



# Program structure



# Program structure



# Player class

```
class Player {
  int xpos, ypos, width, height;
  color playercolor = color(250);

  Player(int screen_x, int screen_y, int pwidth, int pheight)
  {
    xpos=screen_x; ypos=screen_y;
    width=pwidth; height=pheight;
  }
  void move(int x){
    if(x>SCREENX-width) xpos = SCREENX-width;
    else xpos=x;
  }
  int x(){
    return xpos;
  }
  int y(){
    return ypos;
  }
  void draw()
  {
    fill(playercolor);
    rect(xpos, ypos, width, height);
  }
}
```

Why is it better to do this?

# Changes to main program

```
Alien theAliens[];
Player thePlayer;
PImage normalImg, explodeImg;
void settings(){
  size(SCREENX, SCREENY);
}
void setup(){
normalImg= loadImage("invader.GIF");
  explodeImg = loadImage("exploding.GIF");
  theAliens = new Alien[10];
  thePlayer = new Player(SCREENX/2, SCREENY-
  PLAYERHEIGHT-MARGIN,
  PLAYERWIDTH,PLAYERHEIGHT);
  init_aliens(theAliens,normalImg, explodeImg);
}
```

# Changes to main draw method

```
void draw(){
  background(0);
  thePlayer.move(mouseX);
  thePlayer.draw();
  for(int i=0; i<theAliens.length; i++){
    theAliens[i].move();
    theAliens[i].draw();
   }
}
```

# Bullet class

```
class Bullet {
  int width, height, x, y;
  color bulletColor = color (200, 0, 200);

  Bullet(int xpos, int ypos){
    x=xpos; y=ypos;
    width=3; height=6;
  }

  int x(){
    return x;
  }

  int y(){
    return y;
  }

  void move()
  {
    y-=2;
  }

  void draw(){
    fill(bulletColor);
    rect(x, y, width, height);
  }
```

# Collide method

```
void collide(Alien theAliens[]){
   for(int i=0; i<theAliens.length; i++){
     if(x > theAliens[i].x()
      && x < theAliens[i].x()+theAliens[i].width()
           && y > theAliens[i].y()
  && y <theAliens[i].y()+theAliens[i].height()){
        theAliens[i].die();
        return;
      }
}
```

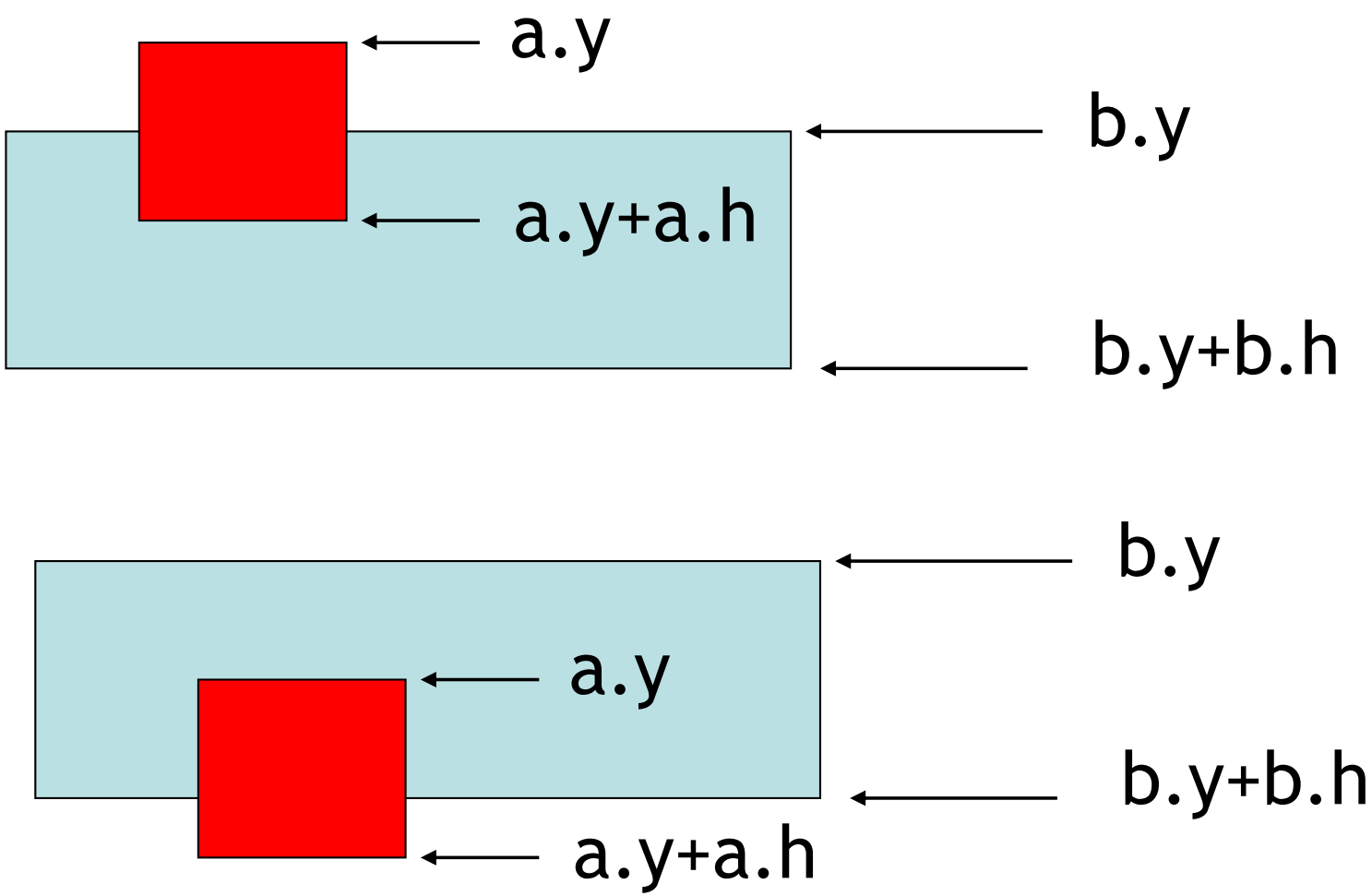
# Changes to main program

```
Alien theAliens[];
Player thePlayer;
Bullet theBullet;
PImage normalImg, explodeImg;

void settings(){
 size(SCREENX, SCREENY);
}

void setup(){
normalImg= loadImage("invader.GIF");
  explodeImg = loadImage("exploding.GIF");
  theAliens = new Alien[10];
  theBullet = null;
  thePlayer = new Player(SCREENX/2, SCREENY- PLAYERHEIGHT-
   MARGIN, PLAYERWIDTH, PLAYERHEIGHT);
    init_aliens(theAliens,normalImg, explodeImg);
}
```

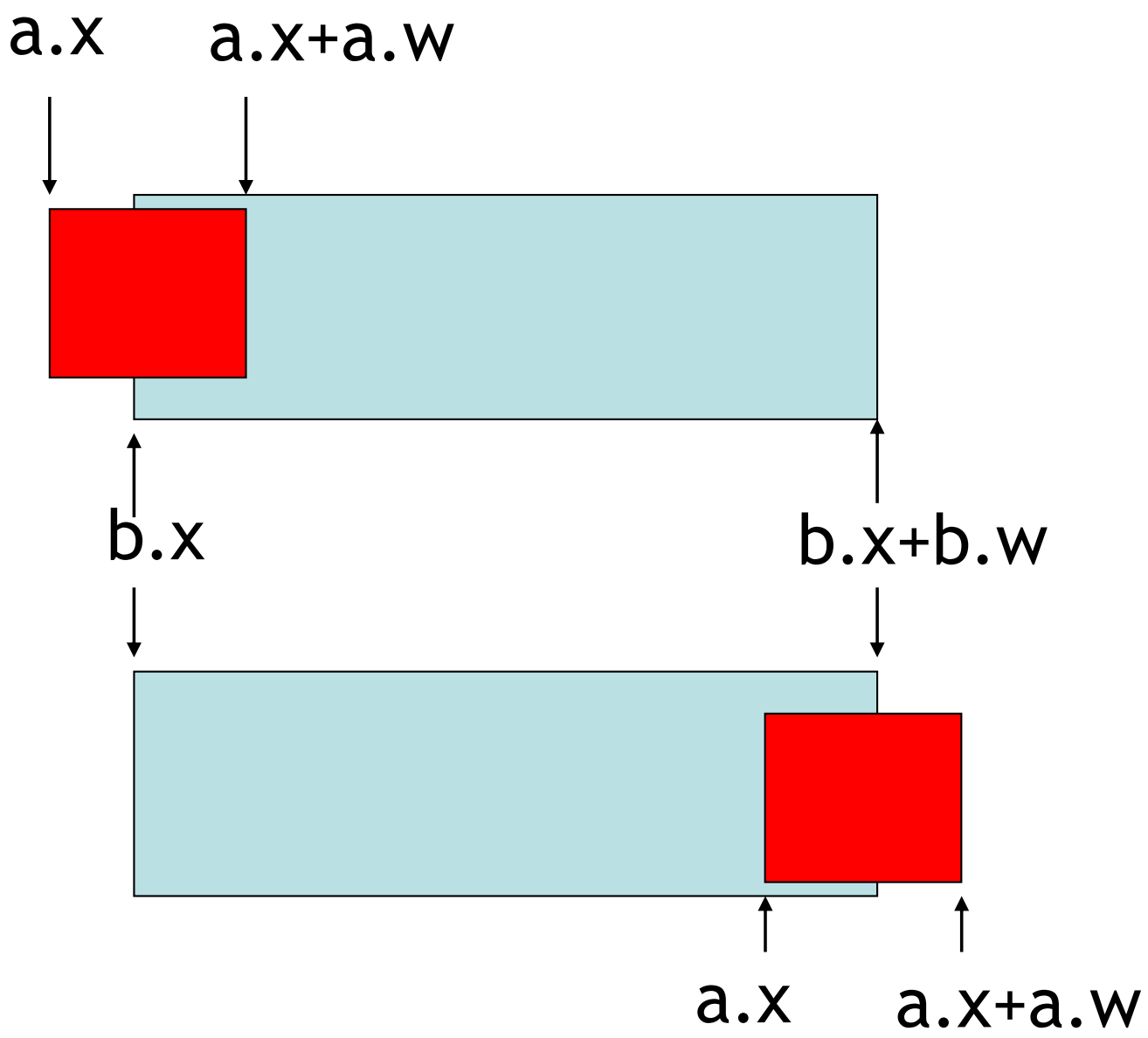
# Changes to main program

```
void draw(){
  background(0);
  thePlayer.move(mouseX);
  thePlayer.draw();
  if(theBullet != null){
    theBullet.move();
    theBullet.collide(theAliens);
    theBullet.draw();
  }
  for(int i=0; i<theAliens.length; i++){
    theAliens[i].move();
    theAliens[i].draw();
  }
}
void mousePressed()
{
    theBullet= new Bullet(thePlayer.x()+thePlayer.width()/2,
   thePlayer.y());
}
```

# Better collision detection



# Better collision detection



# Lists

- Lists allow us to store arbitrary numbers of objects – so don't have to decide exactly how many items we will have.
- Java (and processing) provide an `ArrayList` class which is a type of list.
- Can store any type of object.
- Use `add()` to add an item to the `ArrayList`.
- Use `get(index)` to retrieve an item from the list.
- Need to cast the object returned to the appropriate type before you use it.
  - `myVariable = (myType)myArrayList.get(i);`

# ArrayList of Strings

```
ArrayList myStrings;
void setup(){
  myStrings = new ArrayList();
  myStrings.add("the first string");
  myStrings.add("the second string");
  myStrings.add("the third string");

  for(int i= 0 ; i<myStrings.size(); i++){
    String theString;
    theString = (String) myStrings.get(i);
    println("String number " + i + " is: "+ theString);
  }
}
```

# Exercise – Week 5

1. Building on the ***Alien***, ***Player*** and ***Bullet*** classes from previous weeks, add a ***Bomb*** class:
   - the constructor ***Bomb()*** will create a bomb at a particular x, y position given as arguments.
   - the ***move()*** method will move the bomb one pixel down the screen.
   - the ***draw()*** method will draw the bomb at its current position
   - an **offScreen()** function which will return a boolean value - **true** if the bomb has gone past the bottom of the screen, **false** if it is still onscreen.

- **Demonstrate your code by creating a single instance of Bomb and have it move down the screen.**

# Exercise Week 5

2. Add a ***collide()*** method to **Bomb** which will check whether the bomb has collided with the player (it takes the player as an argument). The method should return a boolean value - **true** if there has been a collision, **false** if there has not been a collision. Alter the main `draw()` method so that if the bomb has collided with the player, a "game over" message is displayed using `text()`.

- **Demonstrate your code by having the bomb go off the bottom of the screen without colliding, and also colliding with the player (with "game over" message).**

# Exercise – Week 5

3. Extend your program so that the aliens that are alive drop bombs. Each Alien will drop at most one bomb at a time. Once a bomb has gone off the bottom of the screen, the alien can drop another bomb.

- One approach is to give each Alien a variable of type ***Bomb***. As part of the move method, the Alien can decide to drop a bomb (ie. create a **new Bomb**), or if it has already got a bomb, it can ***move()*** the bomb. A ***getBomb()*** method returns the bomb if the Alien has one (ie. it is not **null**). This ***getBomb()*** method in Alien is used in the main ***draw()*** method to check whether any of the bombs have collided with the player.

- You might have something like the following inside of the loop which moves all the aliens:

```
Bomb aBomb = theAliens[i].getBomb();
if (aBomb!= null) {
  if (aBomb.collide(thePlayer))
    // bad news for player
  }
```

**Demonstrate your code showing that only the aliens who are alive drop bombs. As before, bombs that do not hit the player should go off the bottom of the screen without colliding, if they collide with the player there should be a "game over" message. If the player kills all the aliens you should print a “you Win” message to the screen (2 Marks)**