

Writing a Java Program (2/2)

EXTRA MATERIAL



covering

- ** using our new programming skills to write a real program (and **learning some new ones on the way!**)
- ** **ArrayList**
- ** Java API (**brief introduction**)



Chapter 6 (*) – “Head First Java” book



These slides are left as **practice and self-study**.

Building the real “Sink a Dot Com” (Recap only)

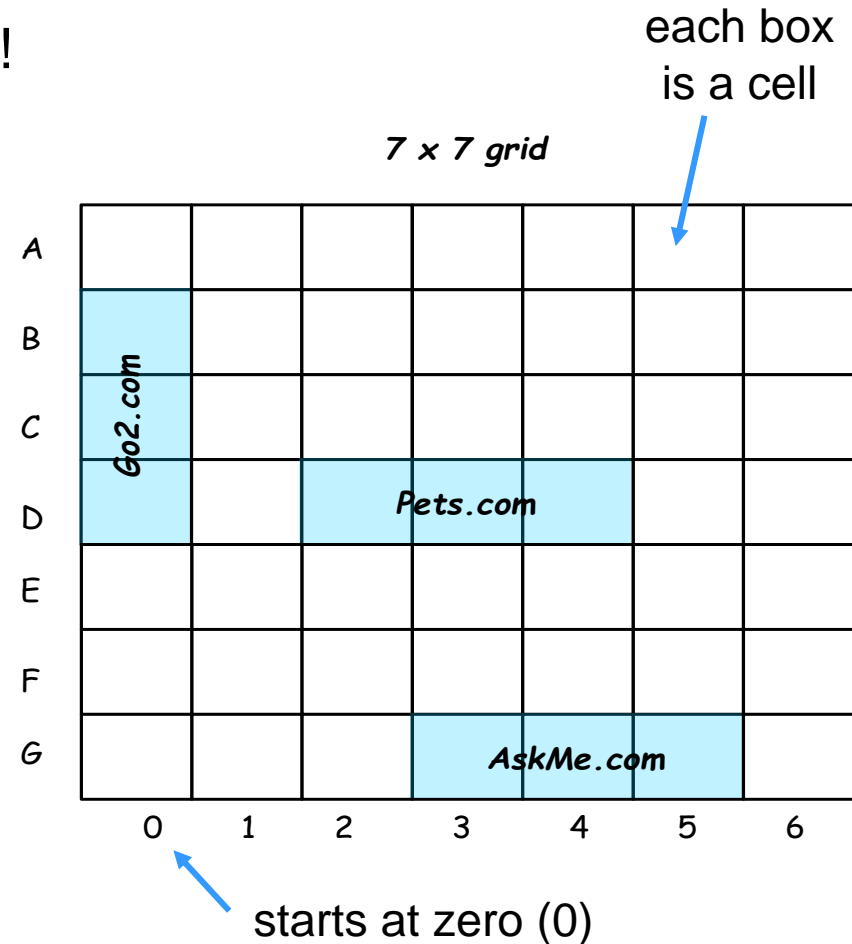
- We have been writing the simple version of the “Dot Com” game.
- Now we have to build the whole thing!

GOAL

- Sink all the computer's “Dot Coms” in the **fewest number of guesses**.
- You are given a **rating**, based on how well you perform.

SETUP

- A **virtual 7x7 board** with **3 randomly placed “Dot Coms”**.
- After that, the player should be prompted to enter their first guess.



What needs to change?

1 DotCom class

Needs a name variable!

Remember that the dot com needs to be able to print its name after being killed! (Ouch! You sunk Pets.com ☹)

2 DotComBust class (the game)

Need 3 **DotComs** instead of 1!

Give each of the **DotComs** a name when created! Need to use a *setter* to do it!

Put the **DotComs** on a grid, rather than a single row. This is a bit complicated so:



to the rescue! Put this in the mysterious **GameHelper** class.

Check the user's guess with ALL 3 of the **DotComs**!

Keep playing until ALL 3 **DotComs** are killed!

Get out of **main()**.

The classes

DotCom

The actual DotCom objects. DotComs know their *name*, *location* and how to check a user guess for a match.

DotComBust

The game class. Makes DotComs and gets user input, plays until all DotComs are dead

GameHelper

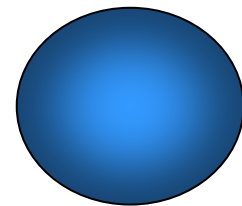
The helper class.
Ready bake...

Can accept user command-line input and make DotCom locations

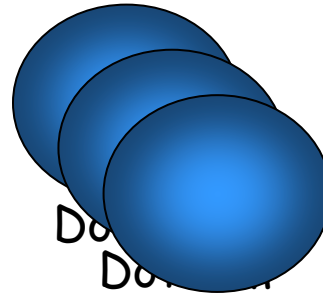
creates and plays with

used for player input and **DotCom** locations

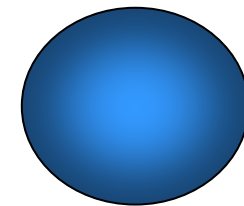
... and the objects



DotComBust



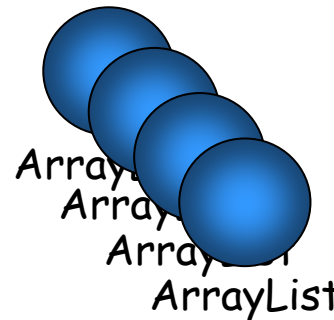
DotCom
DotCom
DotCom



GameHelper



plus 4 ready-baked
objects. Instances of
ArrayLists are objects too!



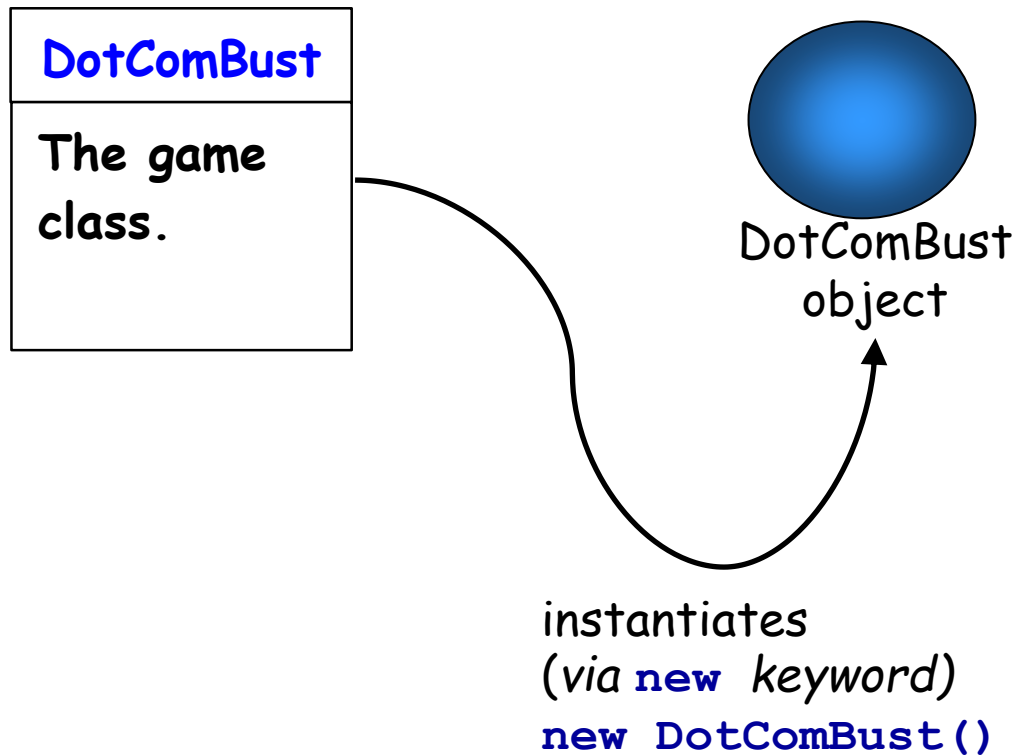
ArrayList
ArrayList
ArrayList
ArrayList

One for DotComBust
One for each instance
of DotCom (*three*)



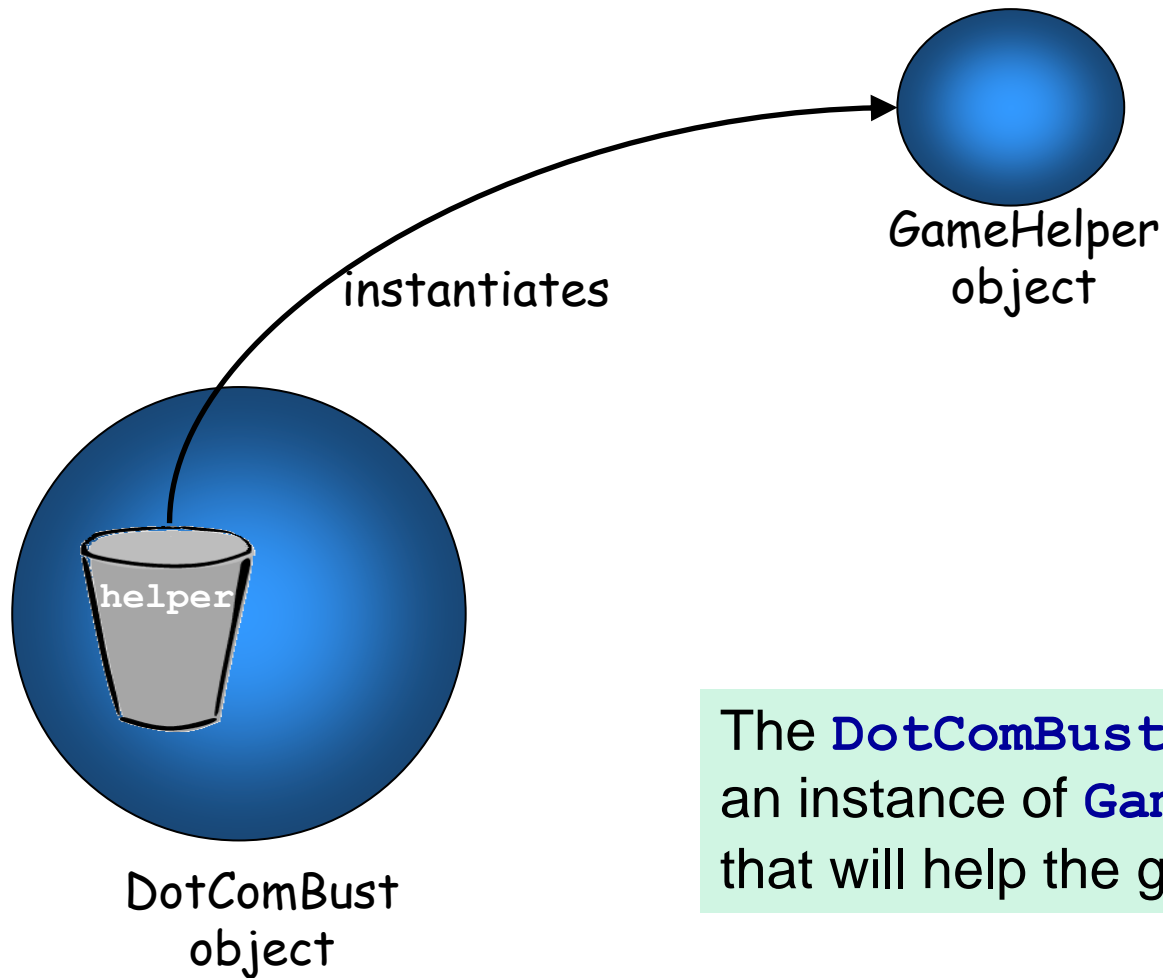
The next slides describe the objects and code necessary to implement a more realistic version of the game, according to [page 2](#).

Who, what, when, where, why?!



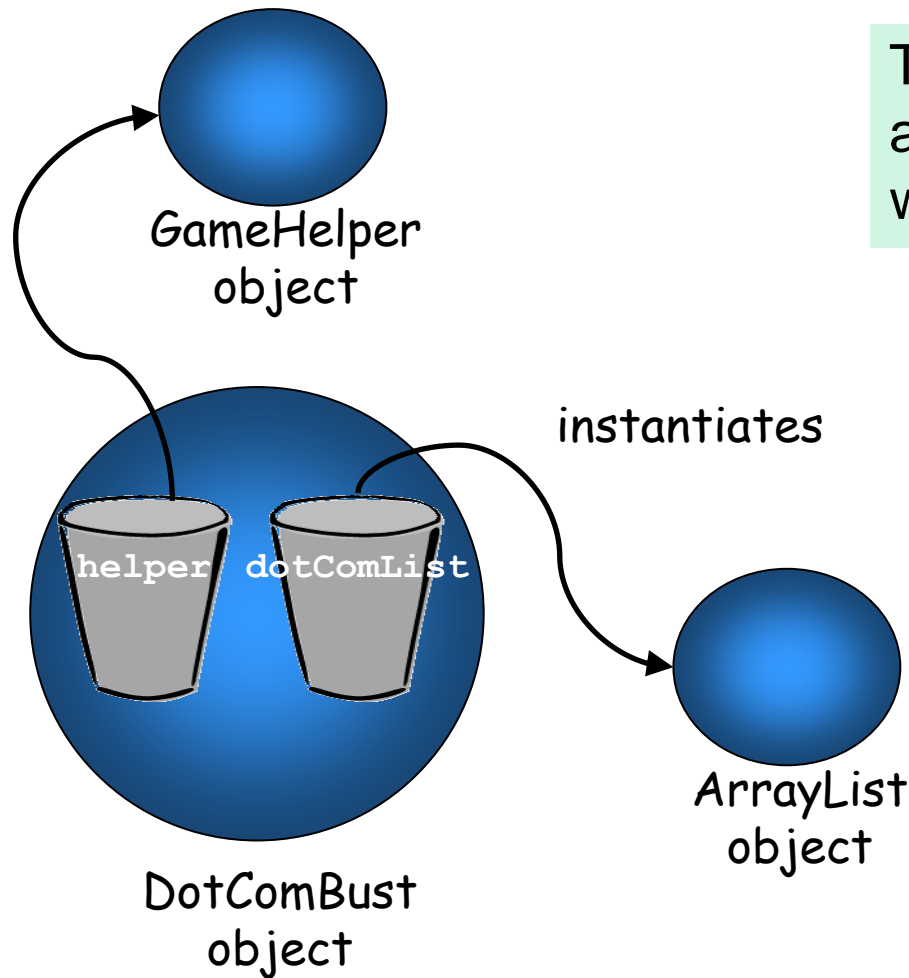
The **main()** method in the **DotComBust** class instantiates the **DotComBust** object that does all the game stuff.

Who, what, when, where, why?! (cont.)



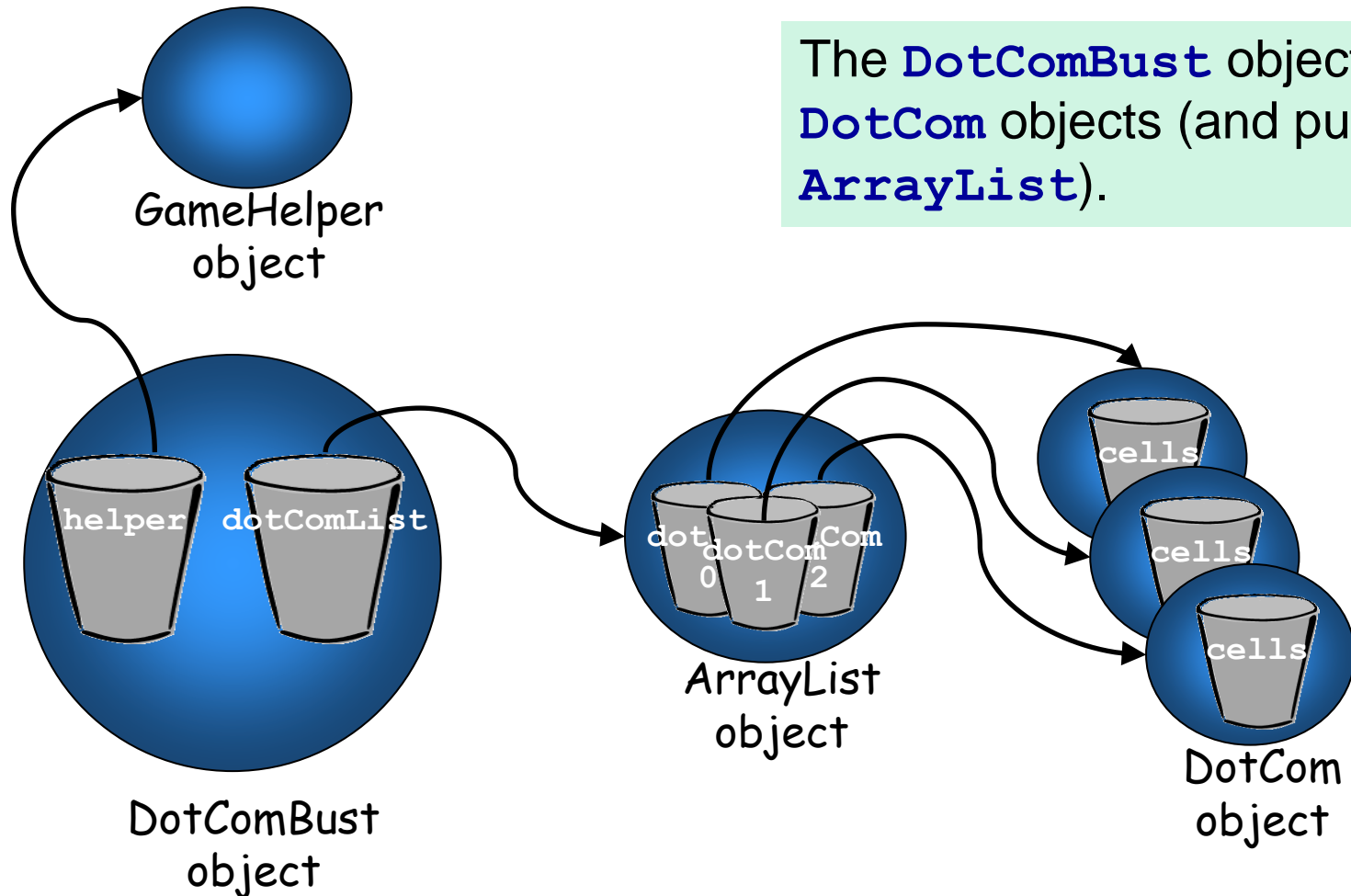
The **DotComBust** object instantiates an instance of **GameHelper**, the object that will help the game do its work.

Who, what, when, where, why?! (cont.)



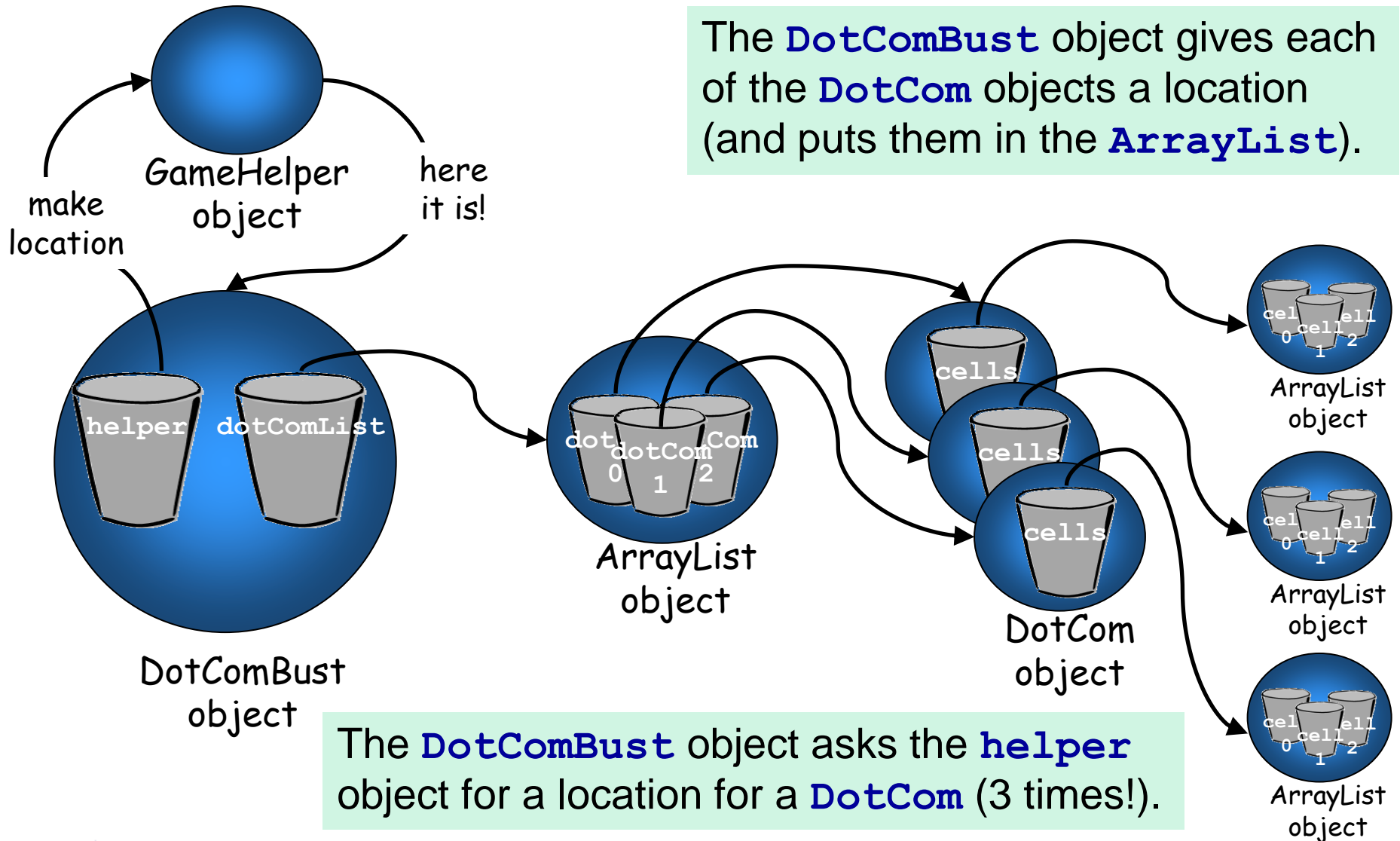
The **DotComBust** object instantiates an instance of an **ArrayList** that will hold the 3 **DotCom** objects.

Who, what, when, where, why?! (cont.)

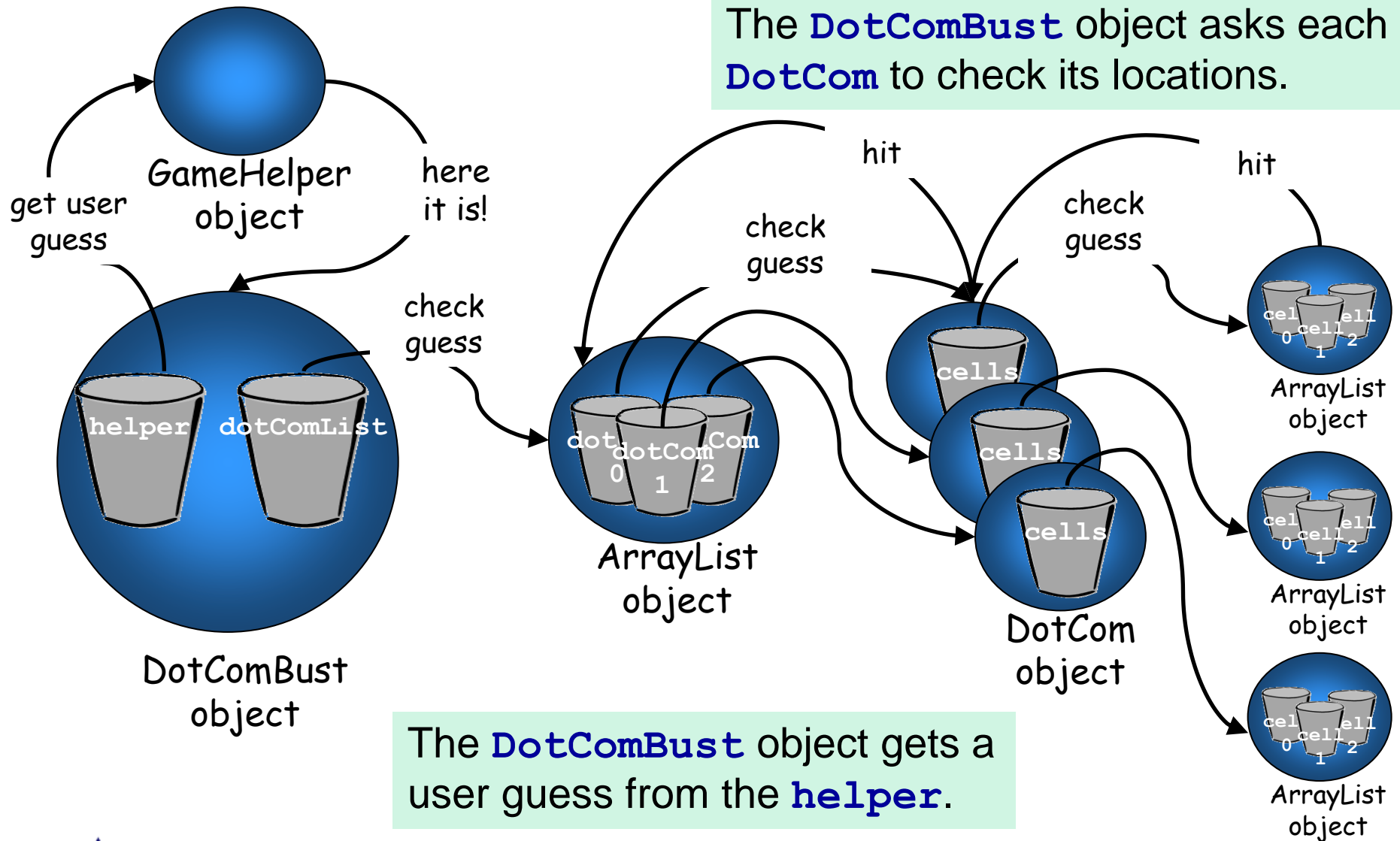


The **DotComBust** object creates 3 **DotCom** objects (and puts them in the **ArrayList**).

Who, what, when, where, why?! (cont.)



Who, what, when, where, why?! (cont.)



The improved code (1/3)

```
import java.util.*;

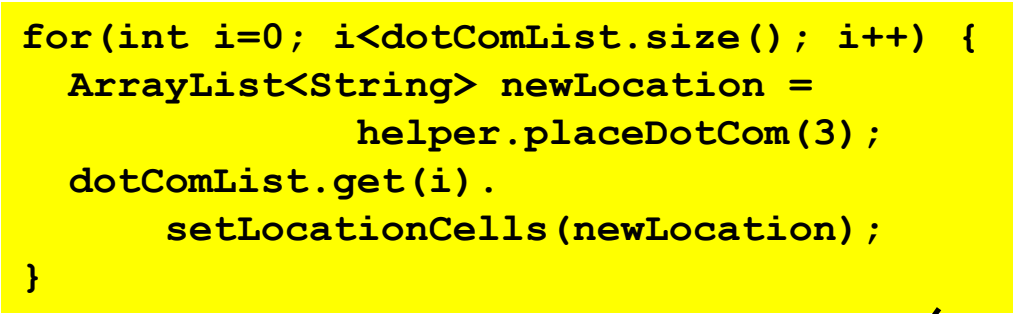
public class DotComBust {
    private GameHelper helper = new GameHelper();
    private ArrayList<DotCom> dotComList = new ArrayList<DotCom>();
    private int numOfGuesses = 0;

    private void setUpGame() {
        // first make some dot coms and give them locations
        DotCom one = new DotCom();
        one.setName("Pets.com");
        DotCom two = new DotCom();
        two.setName("eToys.com");
        DotCom three = new DotCom();
        three.setName("Go2.com");
        dotComList.add(one);
        dotComList.add(two);
        dotComList.add(three);

        for(int i=0; i<dotComList.size(); i++) {
            ArrayList<String> newLocation =
                helper.placeDotCom(3);
            dotComList.get(i).
                setLocationCells(newLocation);
        }

        System.out.println("Your goal is to sink three dot coms.");
        System.out.println("Pets.com, eToys.com, Go2.com");
        System.out.println("Try to sink them all in the fewest number
                           of guesses");

        for (DotCom dc : dotComList) {
            ArrayList<String> newLocation = helper.placeDotCom(3);
            dc.setLocationCells(newLocation);
        }
    }
}
```



```
for(int i=0; i<dotComList.size(); i++) {
    ArrayList<String> newLocation =
        helper.placeDotCom(3);
    dotComList.get(i).
        setLocationCells(newLocation);
}
```

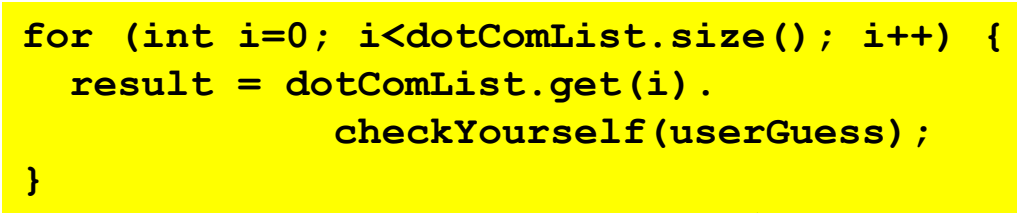
```
for (DotCom dc : dotComList) {
    ArrayList<String> newLocation = helper.placeDotCom(3);
    dc.setLocationCells(newLocation);
}
```

The improved code (2/3)

```
private void startPlaying() {
    while(!dotComList.isEmpty()) {
        String userGuess = helper.getUserInput("Enter a guess");
        checkUserGuess(userGuess);
    }
    finishGame();
}

    for (int i=0; i<dotComList.size(); i++) {
        result = dotComList.get(i).
            checkYourself(userGuess);
    }

private void checkUserGuess(String userGuess) {
    numOfGuesses++;
    String result = "miss";
    for (DotCom dc : dotComList) {
        result = dc.checkYourself(userGuess);
        if (result.equals("hit")) { break; }
        if (result.equals("kill")) {
            dotComList.remove(dc);
            break;
        }
    }
    System.out.println(result);
}
```



The improved code (3/3)

```
private void finishGame() {
    System.out.println("All Dot Coms are dead!
                        Your stock is now worthless.");
    if (numOfGuesses <= 18) {
        System.out.println("It only took you "
                            + numOfGuesses + " guesses.");
        System.out.println("You got out before your options sank.");
    }
    else {
        System.out.println("Took you long enough. "
                            + numOfGuesses + " guesses");
        System.out.println(" Fish are dancing with your options.");
    }
}

public static void main(String[] args) {
    DotComBust game = new DotComBust();
    game.setUpGame();
    game.startPlaying();
}
```