Java API: ArrayList

SimpleDotCom has a Bug!!

```
public class SimpleDotCom {
  private int [] locationCells;
  private int numOfHits = 0;
  public void setLocationCells(int [] locs) { ... }

  public String checkYourself(String stringGuess) {
    int guess = Integer.parseInt(stringGuess);
    String result = "miss";
    for (int cell : this.locationCells) {
      if (guess == cell) {
        result = "hit";
        this.numOfHits++;
        break;
      }
    }
    if (this.numOfHits >= this.locationCells.length) {
      result = "kill";
    }
    return result;
  }
}
```

What will happen when we enter 2, 2, 2?

```
% java SimpleDotComGame
Enter a number   2
hit
Enter a number   2
hit
Enter a number   2
kill
You took 3 guesses.
```

How do we fix it?

- Option 1: Make a second array.

locationCells	4	5	6

hitCells	false	false	false

- You have to
  - check the hitCells array, and
  - change the state.

How do we fix it?

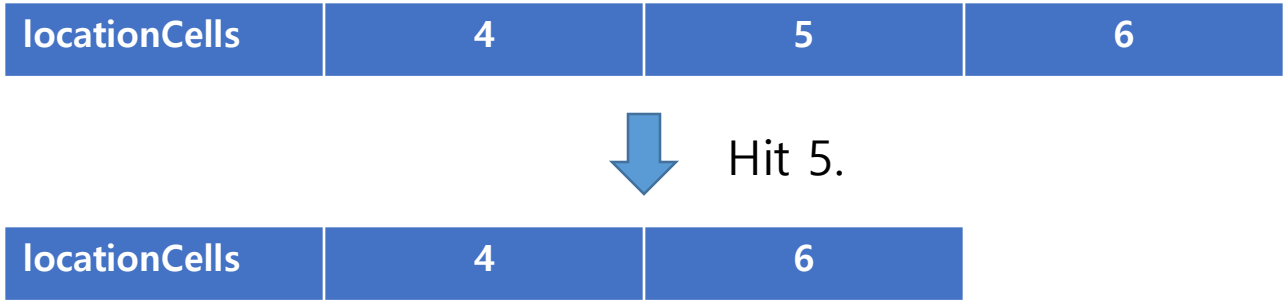
- Option 2: Change the value of any hit cells to -1.

locationCells	4	-1	6

- You have to loop through all three slots even with -1.

How do we fix it?

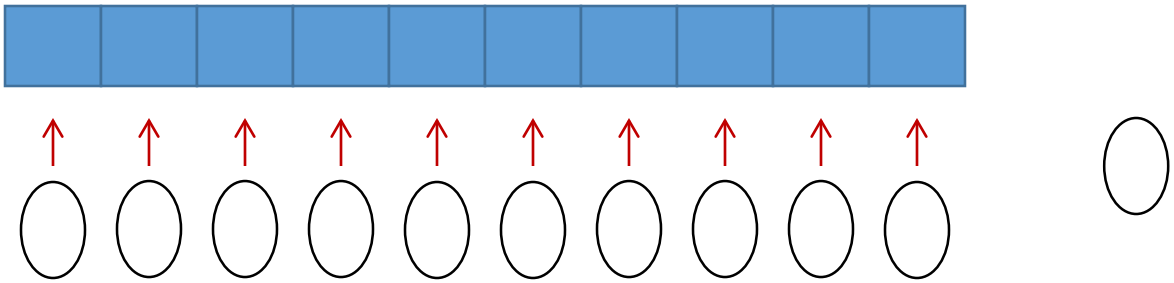
- Option 3: Delete each cell hit by the user.



- You need to shrink the array every time a cell gets hit.

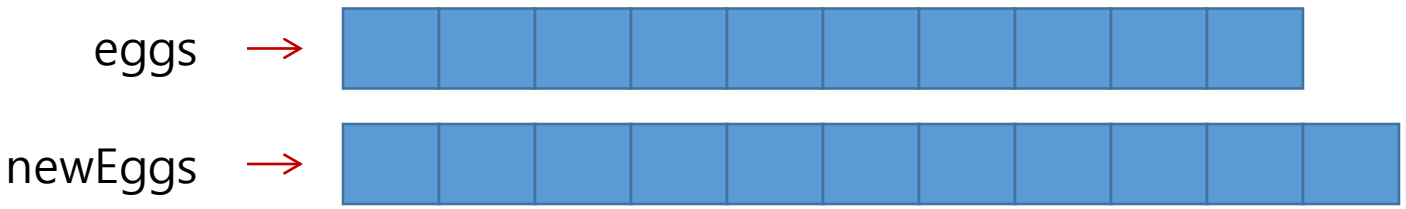
Array (1)

```
Egg[] eggs = new Egg[10]; // Make one

for (int i = 0; i < 10; i++) {
    eggs[i] = new Egg(); // Put eggs in it
}

// I'd like to put an egg in it.
// However, it's full.
```



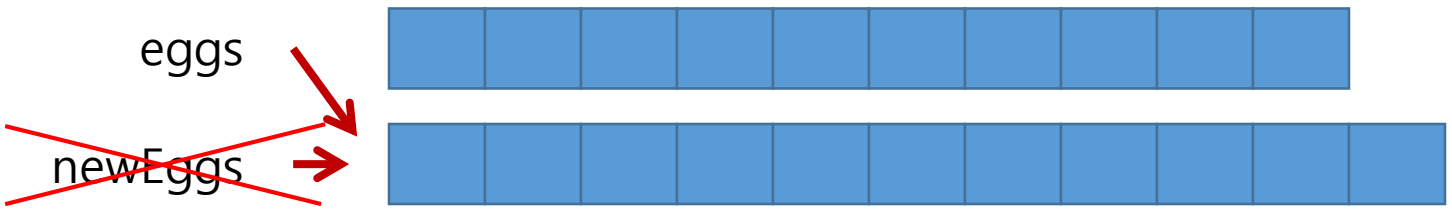
Array (2)

```
Egg[] eggs = new Egg[10]; // Make one

for (int i = 0; i < 10; i++) {
    eggs[i] = new Egg(); // Put eggs in it
}

Egg[] newEggs = new Egg[11];

System.arraycopy(eggs, 0,  newEggs, 0,  eggs.length);

eggs = newEggs;
```



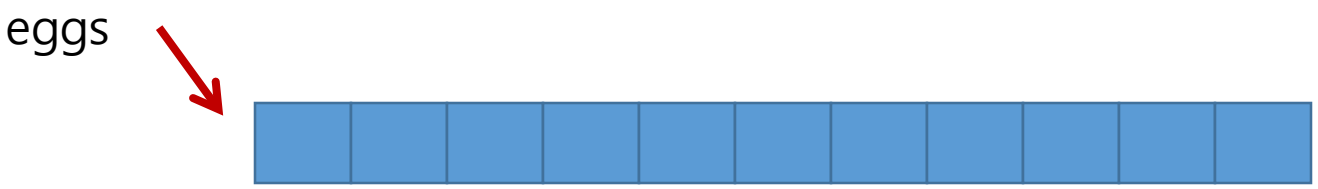
Array (3)

```
Egg[] eggs = new Egg[10]; // Make one

for (int i = 0; i < 10; i++) {
    eggs[i] = new Egg(); // Put eggs in it
}

Egg[] newEggs = new Egg[11];

System.arraycopy(eggs, 0,  newEggs, 0,  eggs.length);

eggs = newEggs;
```



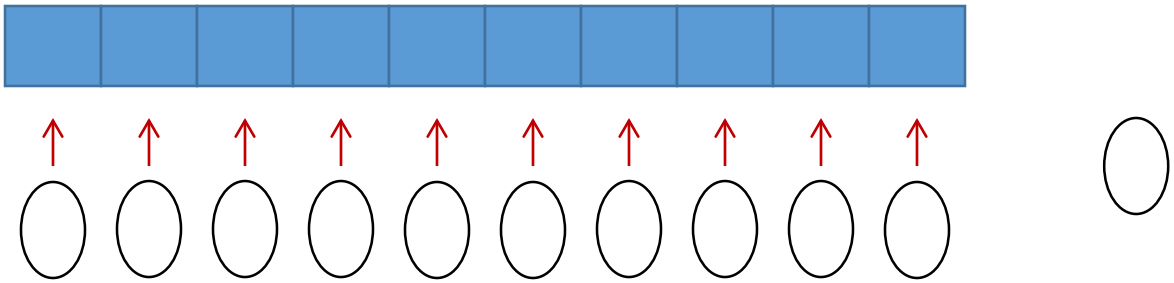
Array (4)

```
Egg[] eggs = new Egg[10]; // Make one

for (int i = 0; i < 10; i++) {
    eggs[i] = new Egg(); // Put eggs in it
}

Egg[] newEggs = new Egg[11];

System.arraycopy(eggs, 0,  newEggs, 0,  eggs.length);

eggs = newEggs;
```



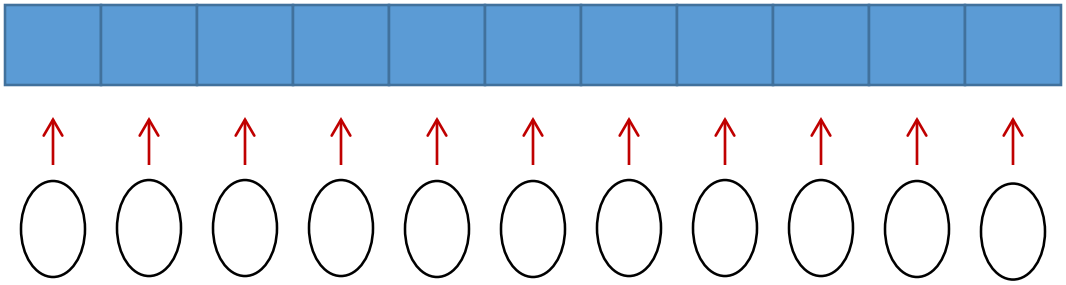
ArrayList<E> (1)

```
ArrayList<Egg> eggList = new ArrayList<Egg>(); // Make one

for (int i = 0; i < 10; i++) {
    eggList.add(new Egg()); // Put eggs in it
}

// I'd like to put an egg in it.
// However, it's full.
```



ArrayList<E> (2)

```
ArrayList<Egg> eggList = new ArrayList<Egg>(); // Make one

for (int i = 0; i < 10; i++) {
    eggList.add(new Egg()); // Put eggs in it
}

// No problem. Just add the new egg.
eggList.add(new Egg());
```



ArrayList<E>: example

```
ArrayList<Egg> eggList = new ArrayList<Egg>(); // Make one

Egg s = new Egg();
eggList.add(s); // Put something in it
eggList.add(new Egg()); // Put another thing in it

int theSize = eggList.size(); // Find out how many things are in it

boolean isIn = eggList.contains(s); // Find out if it contains something

int idx = eggList.indexOf(s); // Find out where something is

boolean empty = eggList.isEmpty(); // Find out if it's empty

eggList.remove(s); // Remove object s from it
eggList.remove(2); // Remove the object at position 2.
```

ArrayList<E>: ArrayListTest.java

```
import java.util.ArrayList;

public class ArrayListTest {
   public static void main(String[] args) {
      ArrayList<Egg> eggList = new ArrayList<Egg>(); // Make one

      Egg s = new Egg("Tommy");
      eggList.add(s); // the eggList has one egg.
      Egg c = new Egg("Cook");
      eggList.add(c) // the eggList has two eggs.
      int theSize = eggList.size(); // the size of eggList is 2.
      boolean isIn = eggList.contains(s); // the eggList contains Tommy.
      int idx = eggList.indexOf(s); // Tommy is the first egg in the list
      boolean empty = eggList.isEmpty(); // It's not empty
      eggList.remove(s); // Remove Tommy from it.
      int idx = eggList.indexOf(c); // Now, Cook is the first egg.
   }
}
```

ArrayList vs. Regular Array

- A regular array has to know its size at the time it's created.

```
String[]  myArray = new String[2];
ArrayList<String> myList =
                        new ArrayList<String>();
```

- To put an object in a regular array, you must assign it to a specific location.

```
myArray[1] = b;
myList.add(b);   or   myList.add(1, b);
```

- Arrays use array syntax that's not used anywhere else in Java.
- ArrayLists are parametrized.
  - `<String>` : Type parameter.

ArrayList<E>: type parameter

- ArrayList<boolean>
- ArrayList<byte>
- ArrayList<char>
- ArrayList<float>
- ArrayList<int>
- ArrayList<long>
- ArrayList<short>

Type parameter must be a class !!
e.g. String, Movie, Dog, etc.

Autoboxing and unboxing (1)

- Wrapper classes
  - Sometimes, a Java compiler do 'autoboxing and unboxing' automatically.

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short

Autoboxing and unboxing (2)

- Example

```
Integer x = 3; // Autoboxing.

x += 5; // Autoboxing: x is an Integer object, but 5 is
not an object.

int y = x; // Autounboxing

ArrayList<Integer> arrList = new ArrayList<Integer>();
arrList.add(Integer.valueOf(0));
arrList.add(1); // possible, autoboxing
int x = arrList.get(1).intValue();
int y = arrList.get(0); // possible, autounboxing
```

Packages (1)

- java.util.ArrayList is a full name.
- java.util is a package name.
- ArrayList is a class name.

```
java.util.ArrayList
```

Packages (2)

- import

```
import java.util.ArrayList;
// ArrayList<Dog> a = new ArrayList<Dog>();

import java.util.*;
// ArrayList<Dog> a = new ArrayList<Dog>();

import java.*;
// ArrayList<Dog> a = new ArrayList<Dog>(); // error
// util.ArrayList<Dog> a = new util.ArrayList<Dog>(); // error
```

Packages (3)

- Write import statements before declaring classes, interfaces, and enums.

```
class A {
    // ...
    import java.util.ArrayList; // error
}
import java.util.ArrayList; // error
```

Packages (4)

- import static
  - You can import static variables using 'import static'.
  - Warning: using 'import static' is not recommended.

```
import static java.lang.Math.PI;

// ...
// double two_pi = 2.0 * Math.PI
double two_pi = 2.0 * PI;
```

- Use Math.PI instead of PI.

SimpleDotCom has a Bug!!

```
public class SimpleDotCom {
   private int [] locationCells;
   private int numOfHits = 0;
   public void setLocationCells(int [] locs) { ... }

   public String checkYourself(String stringGuess) {
      int guess = Integer.parseInt(stringGuess);
      String result = "miss";
      for (int cell : this.locationCells) {
         if (guess == cell) {
            result = "hit";
            this.numOfHits++;
            break;
         }
      }
      if (this.numOfHits >= this.locationCells.length) {
         result = "kill";
      }
      return result;
   }
}
```

How to fix SimpleDotCom

```
public class SimpleDotCom {
    private ArrayList<Integer> locationCells;

    public void setLocationCells(ArrayList<Integer> locs) { ... }

   public String checkYourself(String stringGuess) {
      int guess = Integer.parseInt(stringGuess);
      String result = "miss";
       int index = locationCells.indexOf(guess);
       if (index >= 0) {
            locationCells.remove(index);
            result = “hit”;


      }
       If (locationCells.isEmpty()) {
          result = "kill";
      }
      return result;
   }
}
```

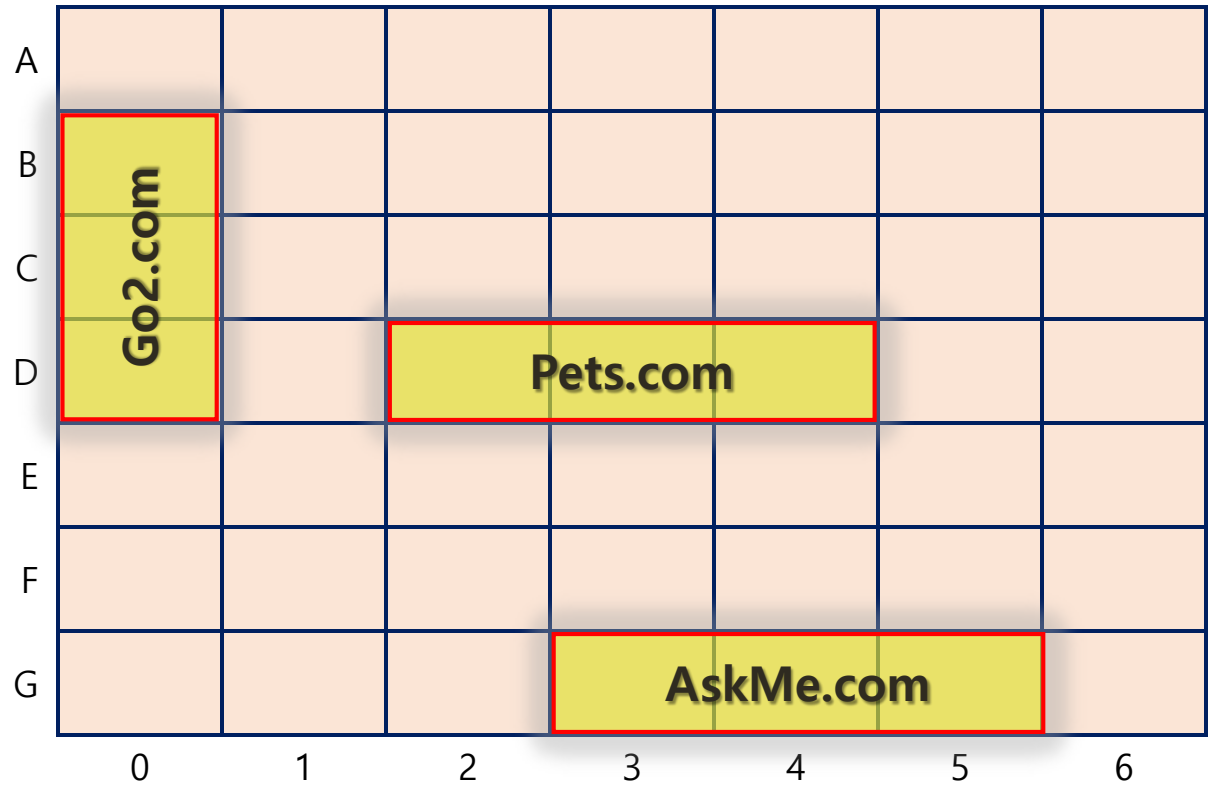
How to fix SimpleDotCom

```
public class SimpleDotCom {
   private ArrayList<String> locationCells;

   public void setLocationCells(ArrayList<String> locs) { ... }

   public String checkYourself(String stringGuess) {

      String result = "miss";
       int index = locationCells.indexOf(stringGuess);
       if (index >= 0) {
            locationCells.remove(index);
            result = “hit”;



      }
       If (locationCells.isEmpty()) {
          result = "kill";
      }
      return result;
   }
}
```

Full Version of DotCom Game

- **Goal**: Sink all of the computer's Dot Coms in the fewest number of guesses.
- **Setup**: When the game program is launched, the computer places three Dot Coms, randomly, on the virtual 7x7 grid. When that's complete, the game asks for your first guess.
- **How you play**: Type at the command-line such as "A3", "C5", etc.

Sink a Dot Com



What needs to change?

- DotCom class (modified from SimpleDotCom)
  - Add a ***name*** variable.
- DotComBust class (modified from SimpleDotComGame)
  - Create *three* DotComs instead of one.
  - Give each of the three DotComs a ***name***.
  - Put the DotComs on a grid rather than just a single row, and do it for all three DotComs.
  - Check each user guess with all three DotComs, instead of just one.
  - Keep playing the game until there are no more live DotComs.
  - Get out of main.

Sink a .Com 1D

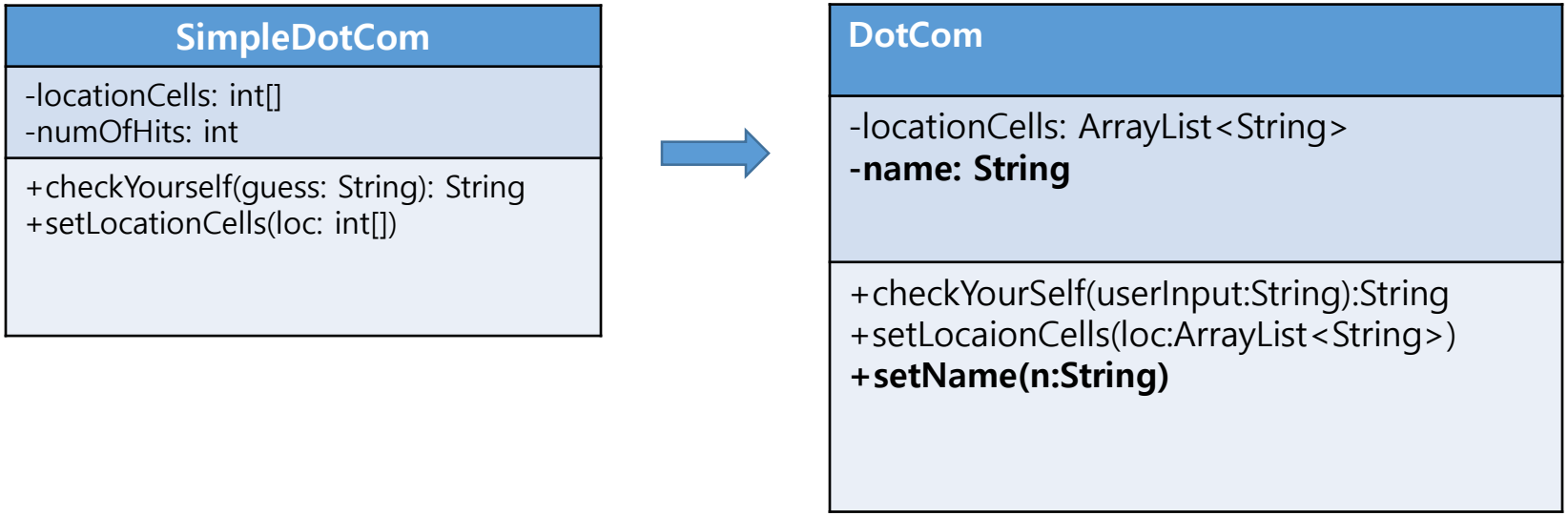
- Simple .Com game

- Class diagram
  - The relationship between the classes will be defined later.

SimpleDotCom
-locationCells: int[] -numOfHits: int
+checkYourself(guess: String): String +setLocationCells(loc: int[])

SimpleDotComGame
+main(args: String[])

DotCom Class



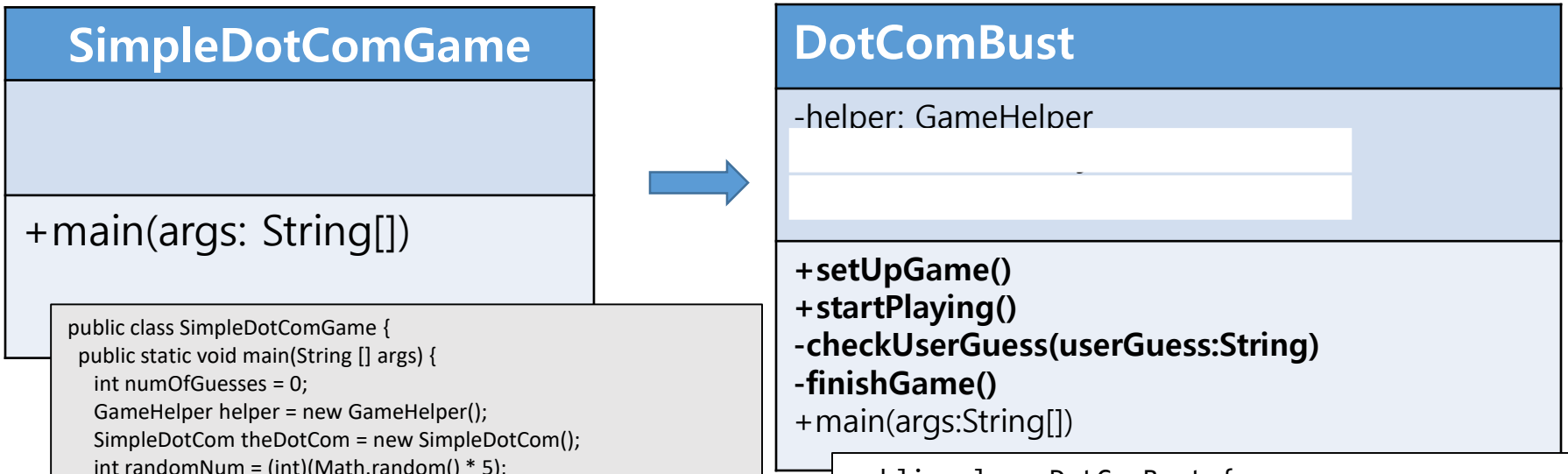
DotCom class (1)

```
import java.util.*;

public class DotCom {
   private ArrayList<String> locationCells
   private String name;

   public void setLocationCells(ArrayList<String> loc) {
      locationCells = loc.clone();
   }

   public void setName(String n) {
      name = n;
   }

   // ...
```

DotCom class (2)

```
   // class DotCom

   public String checkYourself(String userInput) {
      String result = "miss";
      int index = locationCells.indexOf(userInput);
      if (index >= 0) {
         locationCells.remove(index);
         if (locationCells.isEmpty()) {
            result = "kill";
            System.out.println("You sunk " + name + " : ( ");
         } else {
            result = "hit";
         }
      }
      return result;
   }
}
```

DotComBust Class




```
public class SimpleDotComGame {
  public static void main(String [] args) {
    int numOfGuesses = 0;
    GameHelper helper = new GameHelper();
    SimpleDotCom theDotCom = new SimpleDotCom();
    int randomNum = (int)(Math.random() * 5);
    int [] locations = {randomNum, randomNum + 1, randomNum + 2};
    theDotCom.setLocationCells(locations);
    boolean isAlive = true;
    while (isAlive)  {
      String guess = helper.getUserInput("enter a number");
      String result = theDotCom.checkYourself(guess);
      numOfGuesses++;
      if (result.equals("kill")) {
        isAlive = false;
        System.out.println("You took " + numOfGuesses + " guesses");
      }
    }
  }
}
```

```
public class DotComBust {
  ...

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

DotComBust class (1)

```
import java.util.*;

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

  …

}
```

DotComBust class (2)

```
public void setUpGame() {
  DotCom one = new DotCom();     one.setName("Pets.com");
  DotCom two = new DotCom();     two.setName("eToys.com");
  DotCom three = new DotCom();   three.setName("Go2.com");
  dotComsList.add(one);
  dotComsList.add(two);
  dotComsList.add(three);
  for (DotCom dotComToSet : dotComsList) {
    ArrayList<String> newLocation = helper.placeDotCom(3);
    dotComToSet.setLocationCells(newLocation);
  }
}
```

DotComBust class (2)

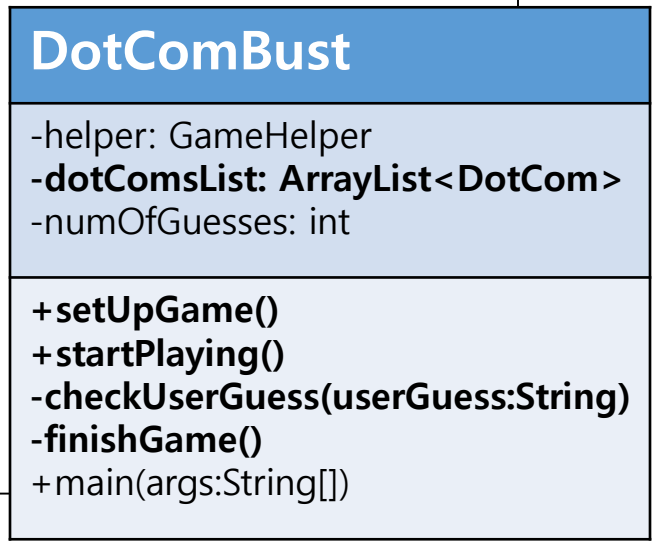
```
public void startPlaying() {
  while(!dotComsList.isEmpty()) {
    String userGuess = helper.getUserInput("Enter a guess");
    checkUserGuess(userGuess);
  }
  finishGame();
}

// ...
```

DotComBust class (3)

```
private void checkUserGuess(String userGuess) {
  numOfGuesses++;
  String result = "";
  for (int x = 0; x < dotComsList.size(); x++) {
    result = dotComsList.get(x).checkYourself(userGuess);
    if (result.equals("hit")) {
      break;
    } else if (result.equals("kill")) {
      dotComsList.remove(x);
      break;
    }
  }
  System.out.println(result);
}
```

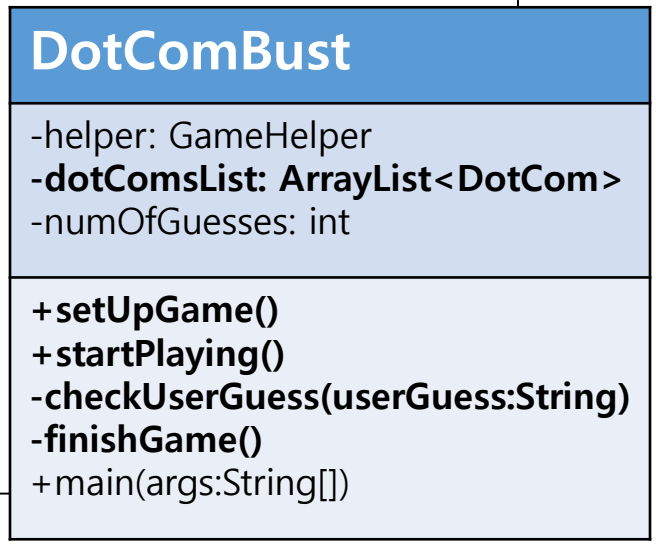
DotComBust class (4)

```
  private void finishGame() {
    System.out.println("All Dot Coms are dead!");
    if (numOfGuesses <= 18) {
      System.out.println(
          "It only took you " + numOfGuesses + " guesses.");
    } else {
      System.out.println(
          "Took you long enough. " + numOfGuesses + " guesses.");
    }
  }

  public static void main(String[] args) {

        setUpGame();
        startPlaying();
  }
}
```

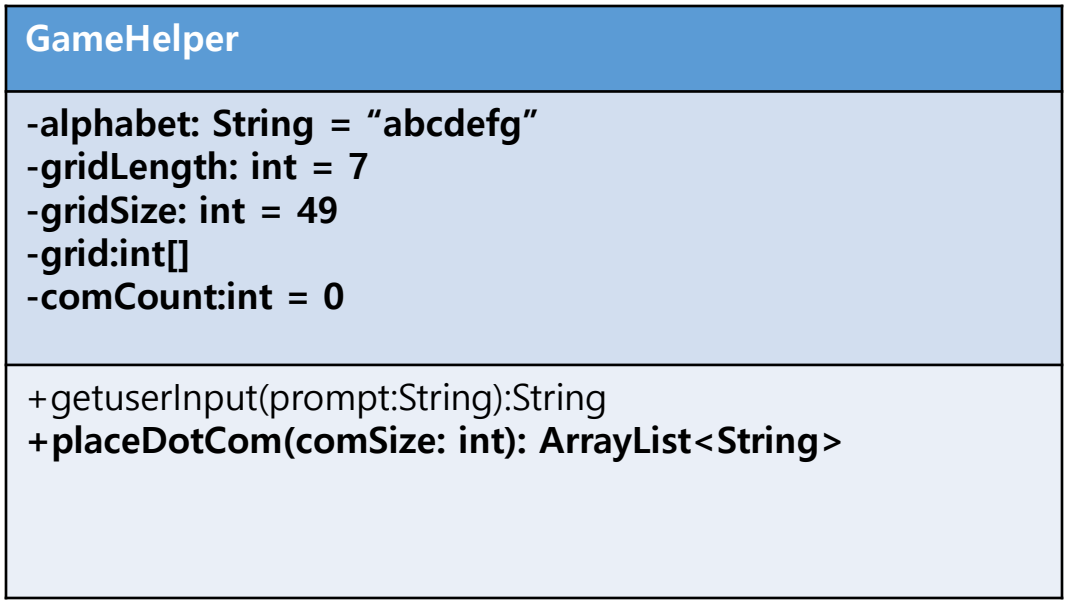


DotComBust class (4)

```
  private void finishGame() {
    System.out.println("All Dot Coms are dead!");
    if (numOfGuesses <= 18) {
      System.out.println(
          "It only took you " + numOfGuesses + " guesses.");
    } else {
      System.out.println(
          "Took you long enough. " + numOfGuesses + " guesses.");
    }
  }

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



GameHelper Classe



GameHelper class (1)

```
import java.io.*;
import java.util.*;

public class GameHelper {
  private static final String alphabet = "abcdefg";
  private int gridLength = 7;
  private int gridSize = 49;
  private int [] grid = new int[gridSize];
  // Alternatively,
  // private int[][] grid = new int[gridLength][gridLength];

  private int comCount = 0;

  // ...
```

GameHelper class (2)

```
// public class GameHelper

public String getUserInput(String prompt) {
  String inputLine = null;
  System.out.print(prompt + " ");
  try {
    BufferedReader is = new BufferedReader(
                                new InputStreamReader(System.in));
    inputLine = is.readLine();
    if (inputLine.length() == 0) return null;
  } catch (IOException e) {
    System.out.println("IOException: " + e);
  }
  return inputLine.toLowerCase();
}

// ...
```

GameHelper class (3)

```
// public class GameHelper

public ArrayList<String> placeDotCom(int comSize) {
  ArrayList<String> alphaCells = new ArrayList<String>();

  int [] coords = new int[comSize];
  int attempts = 0;
  boolean success = false;
  int location = 0;

  comCount++;
  int incr = ((comCount % 2) == 1) ? gridLength : 1;

  // ...
```

GameHelper class (4)

```
…
while (!success && attempts++ < 200) {
  location = (int)(Math.random() * gridSize);
  int x= 0;
  success = true;
  while (success && x < comSize) {
    if (grid[location] == 0) {
      coords[x++] = location;
      location += incr;
      if (location >= gridSize
        || (x > 0 && (location % gridLength == 0))) {
        success = false;
      }
    } else {
      success = false;
    }
  }
}
…
```

GameHelper class (4)

```
…
while (!success && attempts++ < 200) {
  location = (int)(Math.random() * gridSize);
  int x= 0;
  success = true;
  while (success && x < comSize) {
    if (grid[location] == 0) {
      coords[x++] = location;
      location += incr;
      if (location >= gridSize ||
            (x > 0 && incr == 1 && (location % gridLength == 0))) {
        success = false;
      }
    } else {
      success = false;
    }
  }
}
…
```

	0	1	2	3	4	5	6
A	0	1	2	3	4	5	6
B	7	8	9	10	11	12	13
C	14	15	16	17	18	19	20
D	21	22	23	24	25	26	27
E	28	29	30	31	32	33	34
F	35	36	37	38	39	40	41
G	42	43	44	45	46	47	48

GameHelper class (5)

```
    …
    int x = 0;
    while (x < comSize) {
      grid[coords[x]] = 1;
      int row = coords[x] / gridLength;
      int column = coords[x] % gridLength;
      String temp = String.valueOf(alphabet.charAt(column));

      alphaCells.add(temp.concat(Integer.toString(row)));
      x++;
    }

    return alphaCells;
  }
}
```

References

- Kathy Sierra and Bert Bates, *Head First Java*, O'Reilly, 2005.
- Java Platform, Standard Edition 7 API Specification
  - http://docs.oracle.com/javase/7/docs/api/
  - class ArrayList<E>
    - http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Q&A