Objectives Students will be able to…

* **Construct** code using ArrayList.
* **Predict** the output of methods that take arrays as parameters and/or return arrays.

Assessments Students will...

* **Evaluate** statements and **predict** output during a game of grudgeball

Homework Students will...

* **Outline** Chapter 7 and HW 10.1 “ArrayList”
* **Complete** self-check questions #3-6 and exercise #3

# Materials & Prep

* **Projector and computer** (optional)
* **White paper** **and** **markers**
* **Rules** for grudgeball (see website for details: <http://toengagethemall.blogspot.com/2013/02/grudgeball-review-game-where-kids-attack.html>)
* **Team assignments** that divide your class into 5 or 6 teams
* **Nerf hoop & ball** (or wastepaper and trash can)
* **Taped 2- and 3-point lines**

Briefly review the rules of Grudgeball if you have forgotten them. If you have removed your 2 and 3 point lines from last time you played, test out your 2 and 3 point lines before class begins.

|  |  |
| --- | --- |
| Section | Total Time |
| Bell-work and attendance | 5min |
| Introduction and note-taking | 15min |
| Grudgeball | 35min |

# Pacing Guide

# Procedure

*To hook your class for today’s material, and if space and whiteboard setup allow, set up the grudgeball “court” and scoreboard before class begins. Remind students that lecture content will be tested during the game.*

## Bell-work and Attendance [5 minutes]

## Introduction and note-taking [10 minutes]

1. Ask students to name some limitations of arrays: shifting values is an ordeal, adding elements requires forming a new, larger array and copying values over, deleting elements leaves empty, unused indexes.

2. Introduce the more flexible ArrayList (be sure to remind students that they need to import java.util.ArrayList):

* + - * Uses arrays to store values (fast random access)
      * The ArrayList class contains methods to make add, remove, and shift values easily.
      * ArrayList takes a type parameter to determine what kind of values it will use as elements:

ArrayList<String> stores a list of Strings.

ArrayList<Point> stores a list of Points.

* If you forget to pass a parameter with the type you want the array to contain, the code won’t execute.

2. Construct an ArrayList of Strings to demonstrate syntax:

ArrayList<String> spongebob = new ArrayList<String>();

* Even though the notation looks a bit different, the syntax is fairly similar to what we’ve used in the past. ArrayList<type> is how we indicate the type—just like you’d use int when declaring a one dimensional or two dimensional array.

ArrayList<String> spongebob = new ArrayList<String>();

* This is the name of your ArrayList. It can be any non-keyword that you want to use.

ArrayList<String> spongebob = new ArrayList<String>();

* Ask students if they can tell you what the new keyword is for (we use the new keyword when constructing an object).

ArrayList<String> spongebob = new ArrayList<String>();

* Whenever you see empty parentheses, it means that you’re not using parameters.

3. Using Poster 4.7, review some of the methods you can use to manipulate ArrayLists. Add some spongebob elements to your ArrayList:

spongebob.add(“Patrick Star”);

spongebob.add(“Squidward Tentacles”);

spongebob.add(“Mr. Krabs”);

spongebob.add(“Pikachu”);

spongebob.add(“Sandy Cheeks”);

* Ask students for suggestions on how to print out this ArrayList, and ask them to predict the output:

System.out.prinln(“Some of the characters on Spongebob are “ + spongebob);

* Students will probably notice that Pikachu is not a character in the Spongebob cartoon; ask them to refer to Poster 4.7 to suggest some code to remove Pikachu from the list:

spongebob.remove(3); // Pikachu is stored at index 3

* Now ask students to add another character from the show to the middle of the list, at index 3:

spongebob.add(3, “Plankton”);

* + The first parameter 3 indicates the target location, and the second parameter “Plankton” indicates the String to be stored there.

[Patrick Star, Squidward Tentacles, Mr. Krabs, Sandy Cheeks]

becomes

[Patrick Star, Squidward Tentacles, Mr. Krabs, Plankton]

4. Briefly review a few other useful ArrayList methods. Students will have an opportunity to practice (and you will have an opportunity to reteach if needed) during Grudgeball, so this can be a quick overview:

**ARRAYLIST METHODS OVERVIEW**

To get an element from the ArrayList and print it:

System.out.println(spongebob.get(3));

To get the number of elements in the ArrayList and print it:

System.out.prinln(spongebob.size());

To add all the elements in the ArrayList:

int sum = 0;

for (int = 0; i < spongebob.size(); i++){

String s = spongebob.get(i);

sum += s.length();

}

System.out.println(“Total of lengths = “ + sum);

* Have students justify your code choices, and ask a student (or students) to trace the code and narrate the steps for the class.

To replace an array element (no shifting):

spongebob.set(3, “Plankton”);

* This would replace Pikachu with Plankton directly, without requiring the shifting of the array.

To clear an array:

spongebob.clear();

* This removes all elements from the list and leaves null values at each index (it’s an empty array now).

## Grudgeball [35 minutes]

1. Divide students into their assigned teams.

2. Review the rules for grudgeball, and have the students repeat the rules back to you.

3. Using the problems listed below (and any you may add, depending on your class’ needs), play grudgeball until a team wins, or until the class period ends.

a. If a class gets the answer wrong, BRIEFLY pause the game to have students offer corrections before moving to the next team’s question.

b. If correction seems to be dragging on, jump in and quickly re-teach using the incorrect answer as your example. It is important to keep the pace going to maintain student interest in the game!

# 

Gudgeball problems & answers have been grouped assuming that you have 6 teams. If you have fewer teams, each “round” will be shifted accordingly, so you may have rounds where different teams are practicing different concepts. Judge each team’s knowledge gaps, and adjust which questions you ask each group accordingly.

**GRUDGEBALL PROBLEMS AND ANSWERS**

Use a type parameter to declare an ArrayList that:

a. Stores a list of Strings 🡪 ArrayList<String>

b. Stores a list of integers 🡪 ArrayList<Integer> (Wrapper class)

c. Stores a list of Points 🡪 ArrayList<Point>

d. Stores a list of doubles 🡪 ArrayList<Double> (Wrapper class)

e. Stores a list of soccer teams 🡪 ArrayList<String>

f. Stores a list of temperatures 🡪 ArrayList<Double> (Wrapper class)

Construct an ArrayList:

g. Called z that stores a list of ints 🡪 ArrayList<int> z = new ArrayList<Integer>();

h. Called list that stores a list of Strings 🡪 ArrayList<String> list = new ArrayList<String>();

i. Called jose that stores a list of Points 🡪 ArrayList<Point> jose = new ArrayList<Point>();

j. Called pokemon that stores a list of Pokémon 🡪 ArrayList<String> pokemon = new ArrayList<String>();

k. Called metroCard that stores the number of metrocard rides each student has left on their card today 🡪 ArrayList<Integer> metroCard = new ArrayList<Integer>();

**GRUDGEBALL PROBLEMS AND ANSWERS CON’T.**

l. Called location that stores the coordinates of different landmarks 🡪ArrayList<Point> location = new ArrayList<Point>();

Add values to an ArrayList:

m. z that stores a list of ints 🡪 z.add(3);

n. list that stores a list of Strings 🡪 list.add(“This is a string.”);

o. jose that stores a list of Points 🡪 jose.add(new Point(0, 5));

p. called pokemon that stores a list of Pokemons 🡪 pokemon.add(“Pikachu”);

q. called metroCard that stores the number of metrocard rides for each student 🡪 metroCard.add(7);

r. called location that stores the coordinates of landmarks 🡪 location.add(new Point(38.47, 109.3));

Working with the ArrayList given, predict the output of the following code:

Sample ArrayList snacks: [Doritos, Takis, Hot Cheetos, Oreos, Twinkies]

s. snacks.add(1, “Animal Crackers”);

t. snacks.set(3, “Potato Chips”);

u. list.add(1, “Animal Crackers”); 🡪 error! ArrayList called snacks, not “list”

v. snacks.remove(0);

w. snacks.remove(5); 🡪 error! nothing at index 5

x. snacks.remove(0);

snacks.remove(1); 🡪 Takis, Hot Cheetos, Oreos, Twinkies (values shift after first removal)

# Accommodation and Differentiation

In ELL classrooms, read the questions aloud in addition to showing the question on the board or projector. Consider distributing a worksheet with the questions on it so students can write down answers during the game.