159 XTENDED

A WEEKLY REVIEW

Review Session 1, CS 149 Review

Slide 1

- Have everyone sit in groups of 4 to 5 as they are coming in.
- Place a small stack of white printer paper at the center of each table.

Slide 2

- “Objects to Objects” is a spin on the popular party game “Apples to Apples”.
- This slide explains how the game will work. Take a moment to explain the rules to the students.
- In each round, one person at the table will act as a judge. Have the students at each table pick who their judge will be for the first round.
- The TAs will administer the round cards to the judge.
- The rest of the students (the players) should take a sheet of paper from the center of the table. Have them write a unique word on the top corner of the paper that they will remember.
- Once one of the TAs says go, the judges will read the prompt off of the round card to the players. As soon as they are finished reading the prompt, the players can start solving the problem on their papers.
- Once the judges have started reading, start the 5-minute timer on the slide.
- The judges have multiple hints on the bottom of the card that they can optionally choose to give the players while they are solving the problem to make it easier.

- At the end of the 5 minutes, the players flip their papers upside down and hand them to the judge.
- The TAs will then work the problem on the board.
- After the problem has been worked by the TAs, the judge will use that solution to determine which of the players had the most correct answer.
- The player who had the most correct answer is given the round card, which acts as a point.

Slide 3

- Hand out the round 1 card to each table's judge and start the round.

Slide 4

- Show the solution to countJMU

```
public int countJMU(String[] words)
{
    int jmuCount = 0;

    for (int i = 0; i < words.length; i++)
    {
        if (words[i].equals("JMU"))
        {
            jmuCount++;
        }
    }

    return jmuCount;
}
```

Slide 5

- Hand out the round 2 card to each table's judge and start the round.

Slide 6

- Ask the students to start listing primitive and reference types before showing the slide.
- Primitive types: boolean, char, byte, int, short, long, double, float
- Reference types: All objects and arrays
- Compare and contrast primitive and reference types.
- Explain when to use == versus equals()

Slide 7

- Talk about why memory diagrams are different for primitive and reference types.

Slide 8

- Hand out the round 3 card to each table's judge and start the round.

Slide 9

- Solve fizzBuzz.
- Point out that the for loop had to start at 1 and go all the way to 100.

```
public void fizzBuzz()
{
    for (int i = 1; i <= 100; i++)
    {
        if (i % 3 == 0 && i % 5 == 0)
        {
            System.out.println("FizzBuzz");
        }
        else if (i % 3 == 0)
        {
            System.out.println("Fizz");
        }
        else if (i % 5 == 0)
        {
            System.out.println("Buzz");
        }
        else
        {
            System.out.println(i);
        }
    }
}
```

Slide 10

- Hand out the round 4 card to each table's judge and start the round.
- Work the problem on the board shortly before going to slide 11.

Slide 11

- When you copy an array like this, you are only copying over the address to the array.
- Both of these variables point to the same array.
- If you change one, you've changed them both.

Slide 12

- In order to actually create a new array, you must declare a new variable *and* fully instantiate a new array, then copy over all of the elements from the old array.

Slide 13

- Hand out the round 5 card to each table's judge and start the round.

Slide 14

- Explain what a constructor is.
- Explain how we can invoke one constructor from another.

Slide 15

- Explain what getter methods are.
- Explain that we only need these when an attribute is private, which in good programming practice, it always should be.

Slide 16

- Explain what setter methods are.
- Explain that we only need these when an attribute is private, which in good programming practice, it always should be.

Slide 17

- Briefly discuss the toString and equals methods.
- toString is meant to give a string representation of any object; this is useful in many circumstances.
- equals is needed to compare two Cars, since we can't use the equal to (==) operator.