159 XTENDED

A WEEKLY REVIEW

CS 149 REVIEW

Please sit at the tables in groups of 4 or 5

JMU CS Presents...

OBJECTS TO OBJECTS

1. Choose a “Judge” at each table for the first round. Everyone else at the table is a player.
2. The judge should read the prompt aloud to the table.
3. The players will have 5 minutes to complete the activity on a blank sheet of paper.
4. The TAs will work a solution to the problem on the board.
5. The judge will then award the round card to whoever had the most correct answer.

ROUND 1

COUNTJMU SOLUTION

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

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

    return jmuCount;
}
```

ROUND 2

PRIMITIVE TYPES

- boolean
- char
- byte
- int
- short
- long
- double
- float

REFERENCE TYPES

- Arrays
- All Objects
  - String
  - Scanner
  - Integer
  - etc.

Primitive types are defined in the language and is named by a reserved keyword. They are the most basic types in Java.

A reference type variable stores an address in memory where the object or class is located.

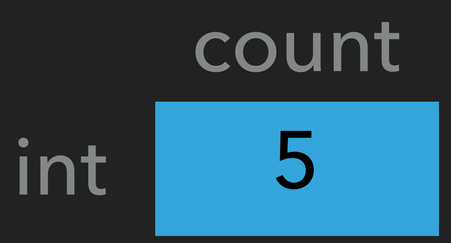
Two primitive types, such as two booleans or ints, can be compared using the equal to operator (==).

Two reference types must be compared using the equals() method.
Using the equal to operator on a reference type compares their location, so it does not work.

TYPES IN MEMORY

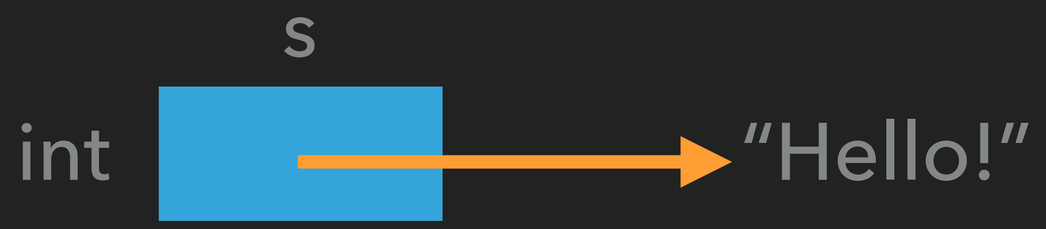
PRIMITIVE

```
int count = 5;
```



REFERENCE

```
String s = “Hello!”;
```



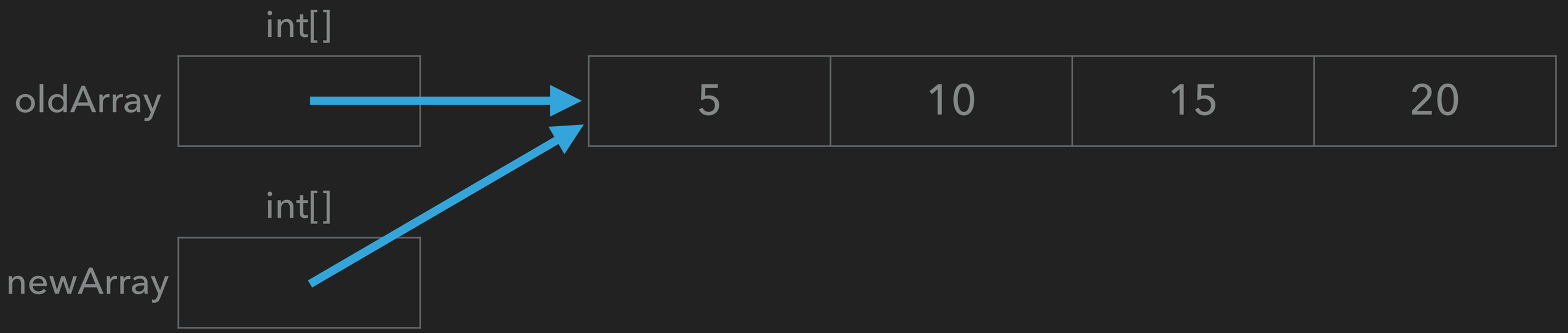
ROUND 3

FIZZBUZZ SOLUTION

```
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);
        }
    }
}
```

ROUND 4

```
int[] oldArray = {5, 10, 15, 20};
int[] newArray = oldArray;
```



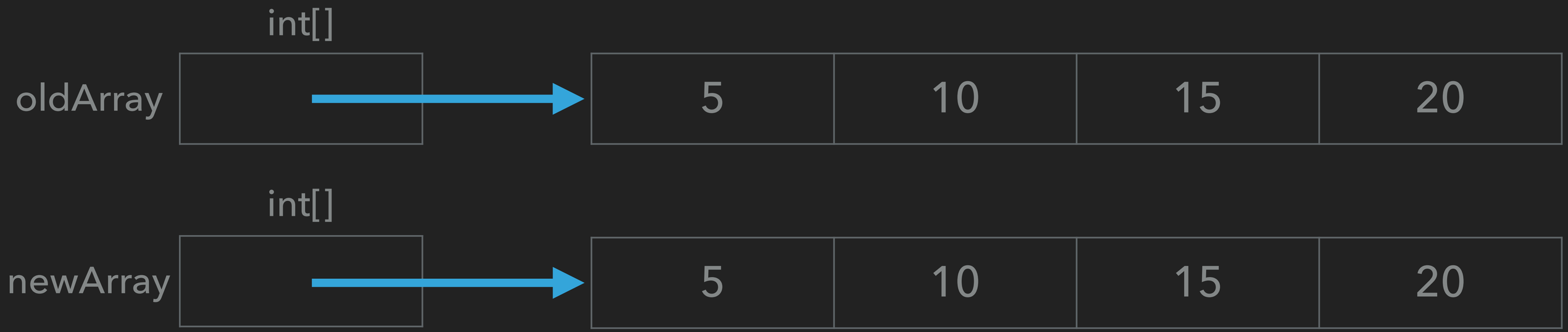
NEWARRAY IS AN ALIAS OF OLDARRAY.

KEY POINT: OLDARRAY AND NEWARRAY POINT TO THE SAME ARRAY.

```
int[] oldArray = {5, 10, 15, 20};

int[] newArray = new int[oldArray.length];

for (int i = 0; i < oldArray.length; i++)
{
    newArray[i] = oldArray[i];
}
```



ROUND 5

CAR SOLUTION

```
public class Car
{
    // Attributes
    private String make;
    private String model;
    private int year;

    public Car(String make, String model, int year)
    {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    public Car(String make, String model)
    {
        this(make, model, 2020);
    }
```

. . .

CAR SOLUTION CONTINUED

```
. . .

    public String getMake()
    {
        return make;
    }

    public String getModel()
    {
        return model;
    }

    public int getYear()
    {
        return year;
    }

. . .
```

CAR SOLUTION CONTINUED

```
. . .

    public String setMake(String make)
    {
        this.make = make;
    }

    public String getModel(String model)
    {
        this.model = model;
    }

    public int getYear(int year)
    {
        this.year = year;
    }

. . .
```

CAR SOLUTION CONTINUED

```
. . .

    public String toString()
    {
        return make + ", " + model + ", " + year;
    }


    public boolean equals(Car other)
    {
        return this.make == other.make
                && this.model == other.model
                && this.year == other.year;
    }

}
```