Please Complete The Socrative Pulse Survey

URL: gosocrative.com

Room Name: JAVAGOLD

Week 1 Overview

Monday	Tuesday	Wednesday	Thursday	Friday
Introduction to Objects and Strings	Collections Part 1	Collections Part 2	Classes and Encapsulation	Review

Objectives for today

1. Objects and Classes
2. Creating Objects
3. Null
4. String Methods

Class-

- A class is a blueprint for an object
- Classes are source code that defines how to create an object (its state and behavior)
- Classes in Java define a datatype

★ Everything in Java is a datatype ★

Objects-

- An in memory data structure that combines state (variables) and behavior (methods)
- Objects exist only in memory while a program is running
- Multiple objects can be created from a class, but each object is distinct and seperate from every other object in a program

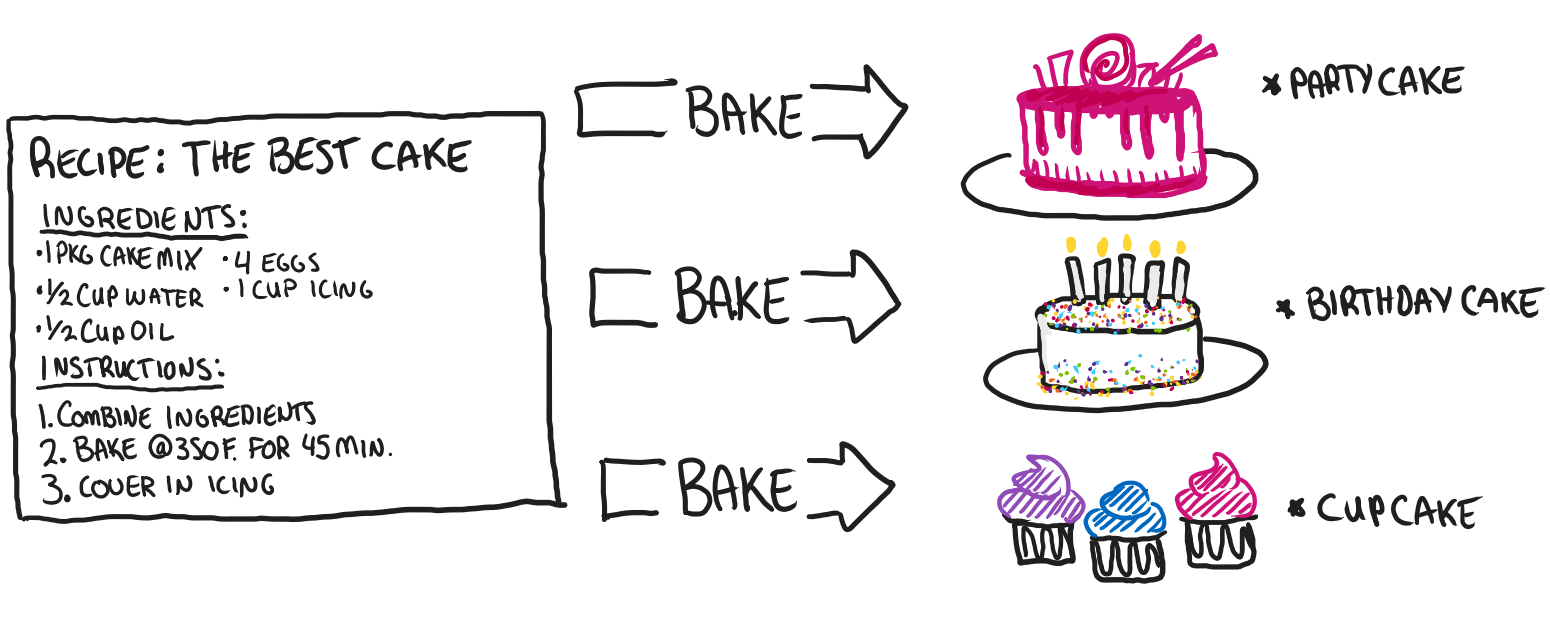
State - The current value of data being stored in an object usually in variables

Memory - A physical device used to store information and programs for immediate use

Behavior - What an object can do. Provided by methods

Abstraction - The hiding of internal implementation details

Class | Objects



Creating Objects

1. Declare a variable to hold the object
2. Instantiate a new object from the class using the new keyword
3. Initialize variables inside the object using setters or constructor

```
Cake partyCake = new Cake();
Scanner in = new Scanner(System.in);
```

```
public class Cake {
// STATE
   private String size;
   private boolean hasSprinkles;

// BEHAVIORS
   public String getSize() {
      return size;
   }
   public void setSize(String size) {
      this.size = size;
   }
   public boolean hasSprinkles() {
      return hasSprinkles;
   }
   public void setHasSprinkles(boolean hasSprinkles)
   {
      this.hasSprinkles = hasSprinkles;
   }
}
```

[NEW ⇒

```
Cake partyCake = new Cake();
partyCake.setHasSprinkles(false);
partyCake.setSize("Medium");
```

[NEW ⇒

```
Cake birthDayCake = new Cake();
birthDayCake.setHasSprinkles(true);
birthDayCake.setSize("Large");
```

[NEW ⇒

```
Cake cupCakes = new Cake();
cupCakes.setHasSprinkles(false);
cupCakes.setSize("Small");
```

Class vs Object

Class
- A template for creating (instatiating) objects within a program
- Logical entity
- Exists only in source code
- Declared using class keyword
- Declared once

Object
- An instance of a class
- Physical entity
- Exists only in memory when program is running
- Created using the new keyword
- Multiple distinct objects can be created using a class

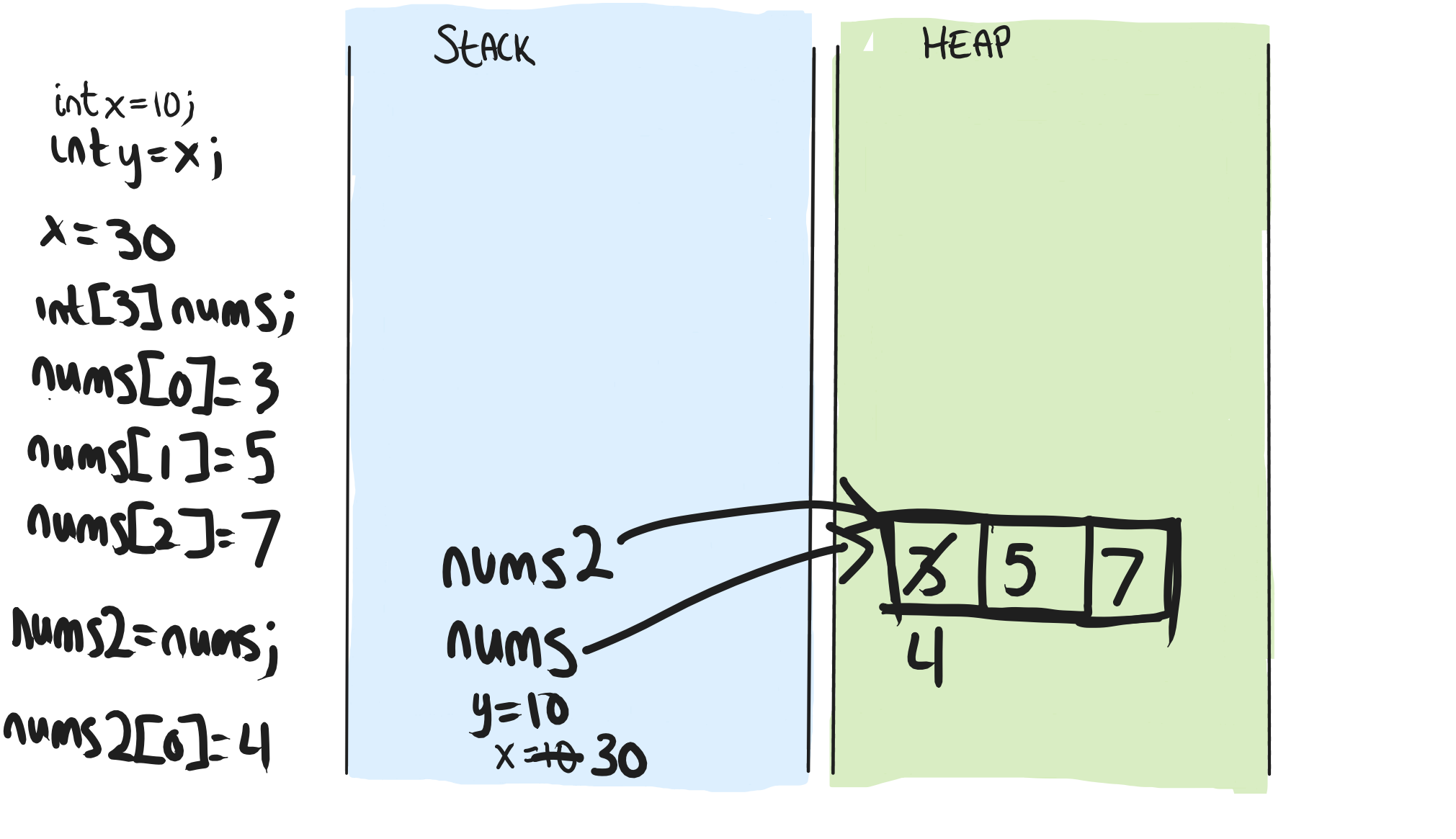
Data Types Memory --- Two types of data types in Java

1. Value type (primitive) reference to a single static space in memory to hold the value. This memory is allocated on the stack.
   - Only 8 primitive types (byte, char, short, int, long, float, double, boolean
2. Reference type (object) does not hold the value in static memory, (stack) but holds a reference to where the object is located in free-floating, dynamic memory called the heap.
   - Objects, Strings, Arrays

Stack Memory vs Heap Memory

Stack Memory
- Linear structure area of a computers memory
- Set size
- Temporary storage
- Fast access
- Managed by OS
- Main problem is running out of memory (stack overflow)

Heap Memory
- Free floating memory area
- Can hold values of any size
- Availiable globally in the application
- Slower access
- Can be resized
- Managed by programmer and language
- Main problem is memory fragmentation (slower access)

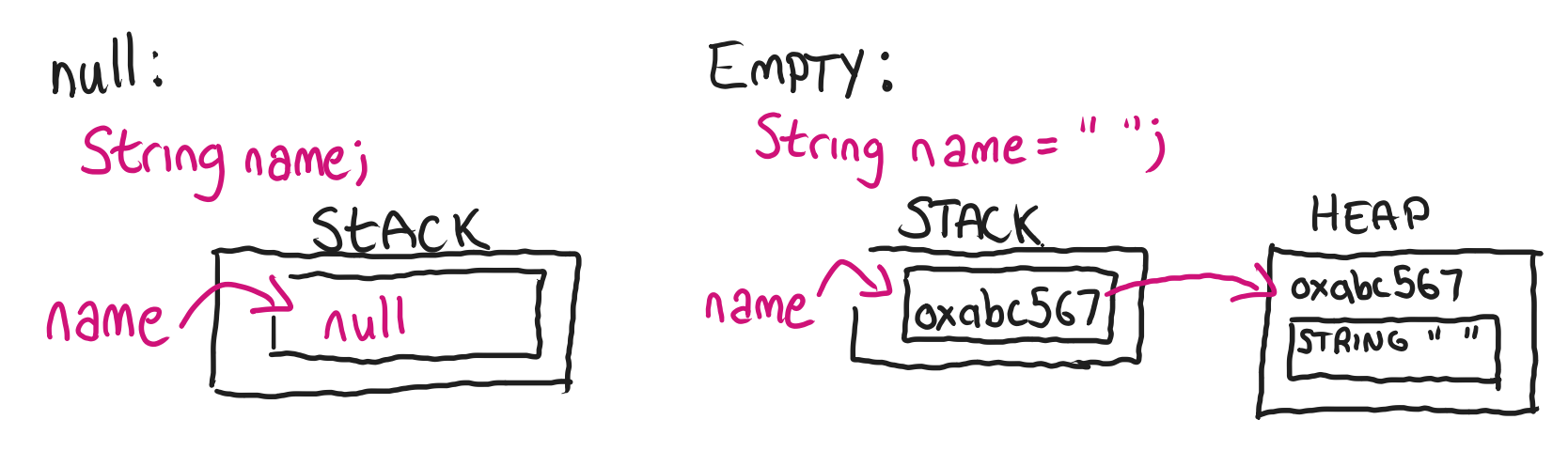


NULL - When variables for reference types are created they default to null, until they are assigned a reference to an instatiated object

! Null is not the same thing as empty !

! Null refers to no value on the stack !

! Empty refers to the value of an existing object on the heap !



Common String Methods

Method	Return Type	Description
contains(string)	boolean	True if the String contains the String passed as an argument.
startWith(string) endsWith(string)	boolean	True if the String starts with or ends with the string passed as an argument
indexOf(string)	int	Returns the starting index of the String passed in the argument
replace(string1, string2)	String	replaces string1 with string2
toLowerCase() toUpperCase()	String	Returns the String in all upper or lower case
split(str)	String[]	Splits the str into an array using the str in the argument as a delimiter
equals(string) equalsIgnoreCase(string)	boolean	True if the value of the string is equal to the string passed in the argument. equalsIgnoreCase() compares the string without case.
trim()	String	removes whitespace from the beginning and ending of the String
substring(startIndex, endIndex)	String	Returns a part of the string from startIndex up to, but not including, endIndex

String Static methods

Static methods are methods that are not available on the objects, but only using the data type.

For example: `String intAsString = String.valueOf( 10 );`

Method	Return Type	Description
valueOf(x)	String	Returns the String representation of x
join(delimiter, list of strings)	String	Returns a new String composed of the strings in the list separated by the delimiter.