CMSC 131

Object-Oriented Programming I

Introduction to Classes II

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Overview

- Heap
- Equals

Class Review

- Let's review what we have discussed so far about classes
- Class vs. Object
- How to define a class
- How to define methods and data in a class
- What is an instance variable
- What is static method
- What are getters and setters
- How to use methods to avoid code duplication

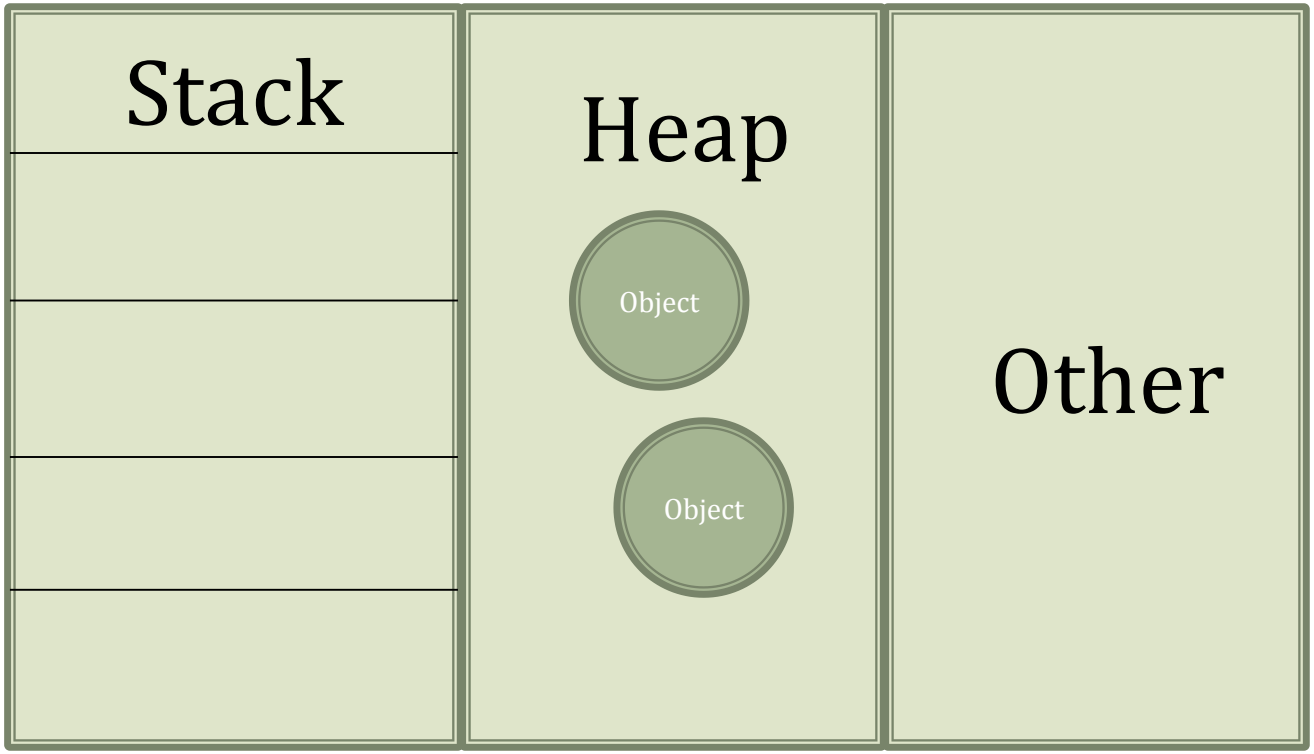
How Are Objects Created?

- In Java: using new

  Recall:

  Scanner sc = new Scanner(System.in);
- Invoking new:
  - Creates an object in a memory area called the “heap”
    Space is created for instance variables
  - Returns the address/reference where the object lives

Main Memory Organization



In Java, 9 Sorts of Variables

- 8 primitive types
  - Types are the 8 built-ins (int, byte, double, etc.)
- Reference type
  - Objects always stored in heap (including all data)
  - Reference to objects are another type, and hold one memory address (typically one word)
- Stack holds local variables
  - e.g. int x
  - e.g. String str; // str is reference variable
- Heap holds allocated memory (i.e., with “new”)
  - e.g. Scanner sc = new Scanner(System.in);

Strings Are Objects

- Where is new in

  String name = “Batman”; ?

- Java provides it!
  - String is special because it is used so often
  - Java automatically “fills in” new
  - You can too:

    String name = new String(“Batman”);

Heap Issues

- What happens if new is called and there is no free heap?
  - Crash!
- What happens if following are executed?

  String s;

  s = new String(“cat");

  s = new String(“dog");

  s = new String(“cow");
- Let’s draw a diagram
- Wasted heap
  - “cat”, “dog” no longer referenced by stack
  - Crashes become a problem!

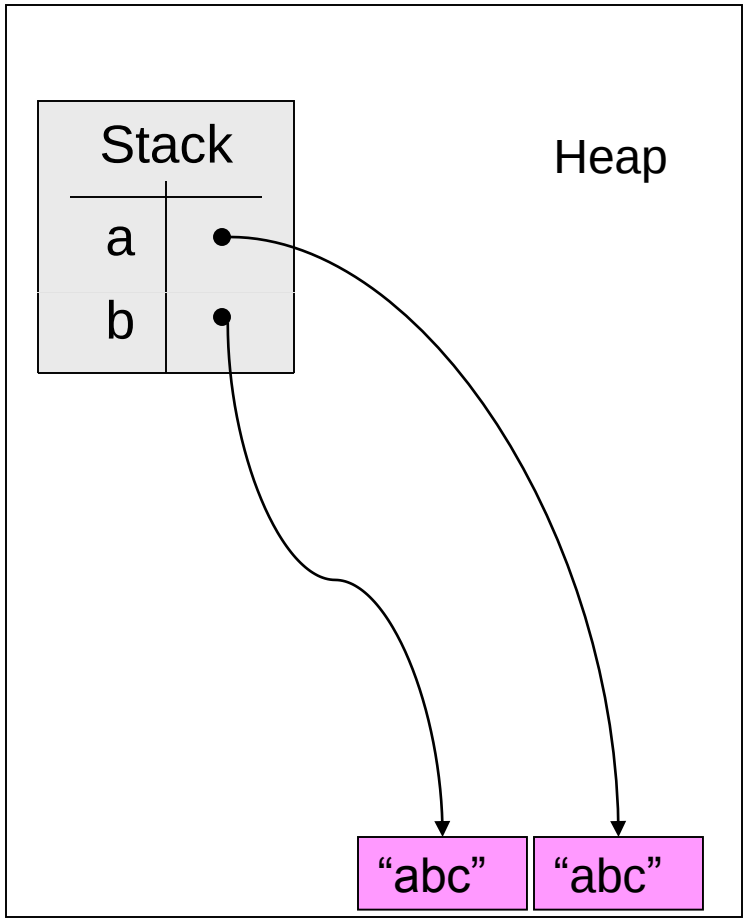
Garbage Collection

- This “heap management” or “memory management” issue is central in CS
- Java copes by invoking the garbage collector to reclaim unused, but still-allocated heap space
- Garbage collector reclaims memory in allocated heap and returns it to free heap
- In previous example, “cat” and “dog” would be reclaimed
- In some languages (e.g., C++) you need to take care of reclaiming memory
  - Use of delete operator in C++ otherwise you will have memory leaks

Example

```
String a = new String (“abc”);
String b = new String (“abc”);

if (a == b) {
    println (“Equal”);
} else {
    println (“Not equal”);
}
    => Not equal
```



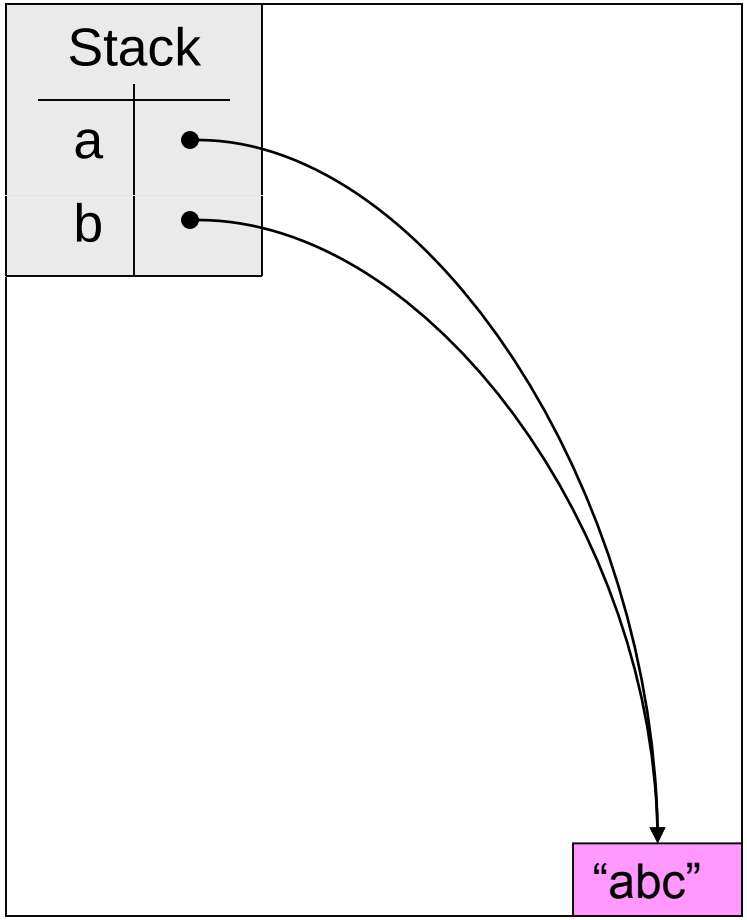
Contrasting Example

```
String a = new String (“abc”);
String b = a;

if (a == b){
   println (“Equal”);
} else {
   println (“Not equal”);
}
```

=> Equal

- This is called ALIASING →Two variables refer to same object.



equals

- **==** checks if two reference variables refer to the same object
- Methods like str.equals() check if two different objects have the same “content”
- Other classes will have an equals method also

Advanced Diagram

- Here is a more advanced diagram for memory organization
  - http://blog.codecentric.de/wp-content/uploads/2009/12/java-memory-architecture.jpg