COMP90041

Programming and Software Development

Semester 1, 2021

Lab 5

Andrew Naughton

andrew.naughton@unimelb.edu.au

THE UNIVERSITY OF
MELBOURNE

Outline

- ❖ Static methods & variables
- ❖ Memory
- ❖ Stack & Heap
- ❖ Primitive vs. Class
- ❖ Pass by value
- ❖ Assignment 1 spec
- ❖ Exercises

THE UNIVERSITY OF
MELBOURNE

Static methods

- A method that belongs to the class, rather than an instance/object of that class
- Declared with:
  - <public | private> static <return type> <method name>(params) {…}
  - E.g. public static void main(String[] args) {…}

THE UNIVERSITY OF
MELBOURNE

Static methods

- Invoked using the class name, rather than the calling object – e.g.
  - returnedValue = MyClass.myStaticMethod(args);
  - Math.max(); Math.round();
  - Compared to non-static methods: keyboard.nextLine();
- Static methods cannot access non-static (instance) variables or non-static methods
- However, they can access other static variables and static methods

THE UNIVERSITY OF MELBOURNE

Static variables

- ❖A variable that belongs to the class, rather than an instance/object of that class
- ❖There is only one copy of a static variable per class, compared to instance variables where each object has its own copy
- ❖All objects of the class can read and change a static variable
- ❖A static variable is declared like an instance variable, but with the modifier static:
  - ❖<public | private> static <return type> <var name>;
  - ❖E.g. private static int myStaticVariable;

THE UNIVERSITY OF MELBOURNE

Memory

- ❖ A computer has two forms of memory: Secondary and Main
- ❖ Secondary memory is used to hold files for "permanent" storage – USB, Disk
- ❖ Main memory is used when running a program – e.g. a Java program :)
- ❖ Main memory consists of a long list of numbered locations called bytes, where each byte is 8 bits (e.g. 10100100)

THE UNIVERSITY OF MELBOURNE

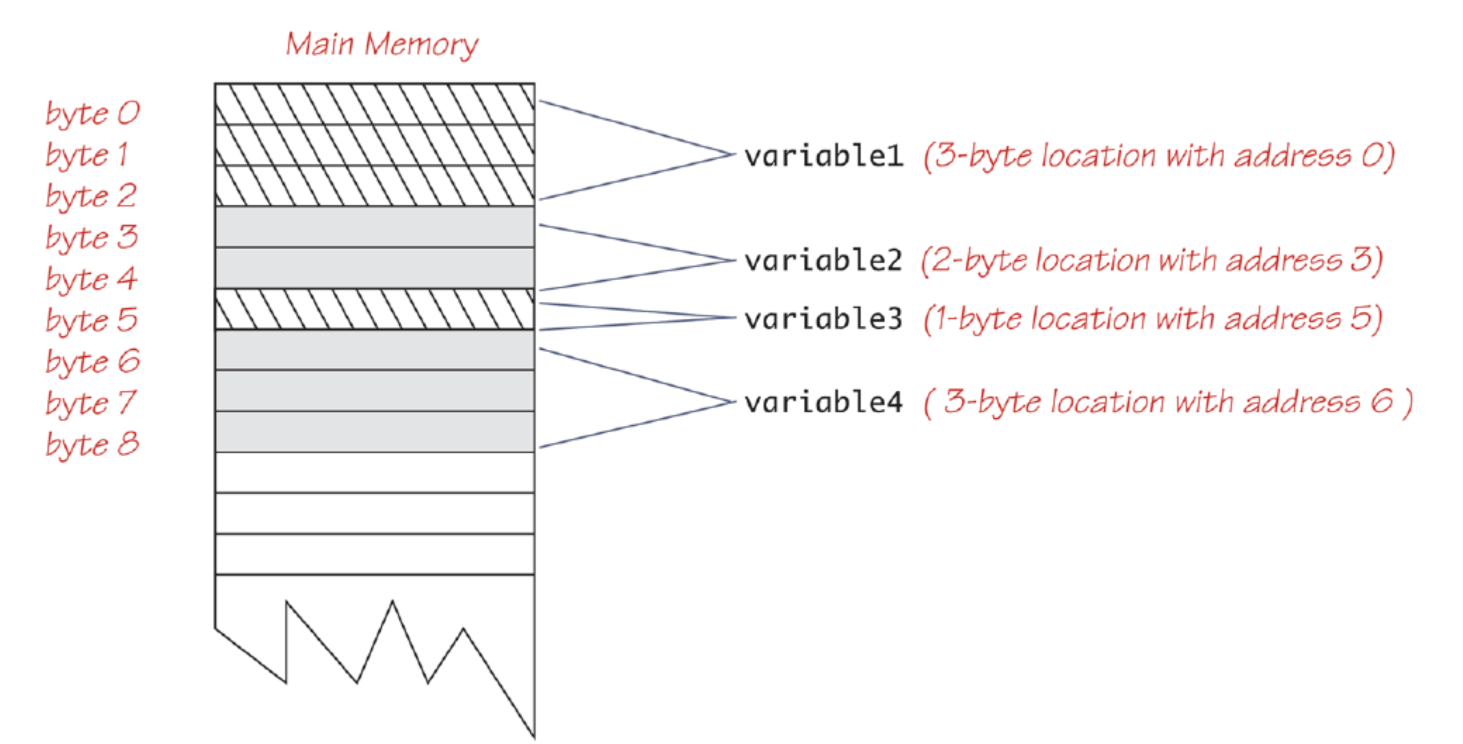
Main Memory

- ❖ The number that identifies a byte is called its address
- ❖ A data item can be stored in one or more of these bytes
- ❖ The address of the byte is used to find the data item
- ❖ Most data types require more than one byte of storage:
  - ❖ In this case, several adjacent bytes are used. The entire chunk of memory is its memory location.
  - ❖ The address of the first byte of this memory is used as the address for the data item

Main Memory

- ❖ A computer's main memory can be thought of as a long list of memory locations of varying sizes

Display 5.10 **Variables in Memory**

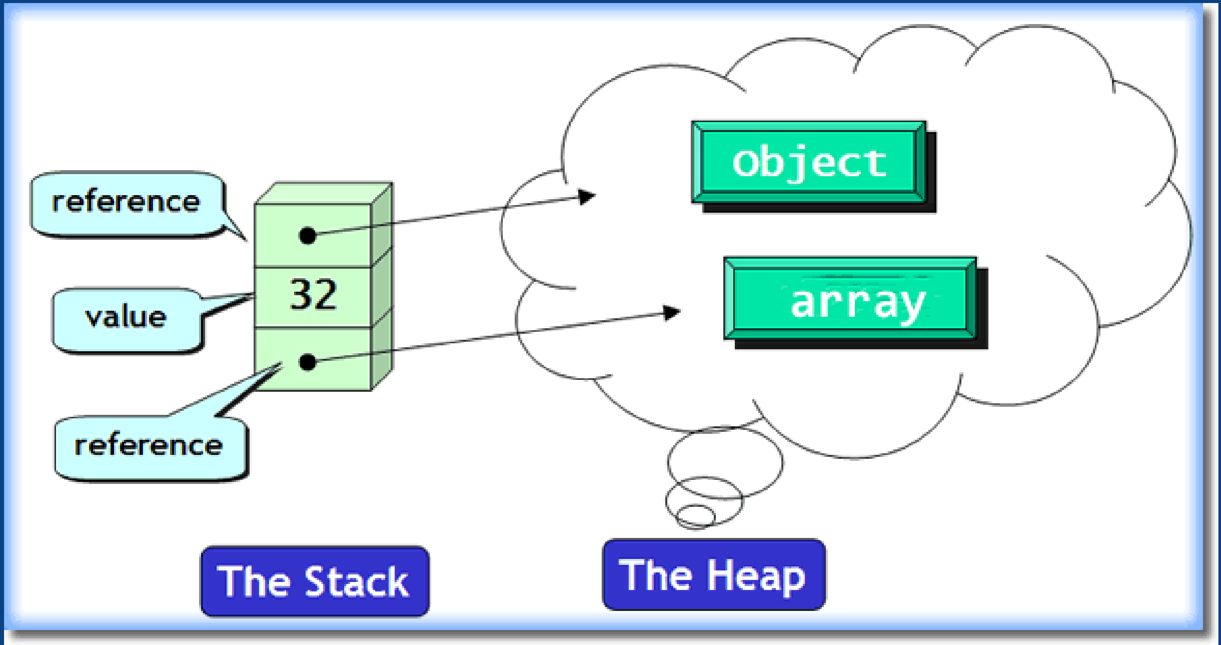


Main Memory
byte 0
byte 1
byte 2
byte 3
byte 4
byte 5
byte 6
byte 7
byte 8
variable1 (3-byte location with address 0)
variable2 (2-byte location with address 3)
variable3 (1-byte location with address 5)
variable4 (3-byte location with address 6)

THE UNIVERSITY OF MELBOURNE

Stack & Heap

- ❖Two types of Main Memory: Stack and Heap
- ❖For primitive typed variables, their values get stored in Stack Memory
- ❖For class-typed variables, their memory address (or reference) where its object is located gets stored in Stack Memory. The object itself is stored in Heap Memory.



THE UNIVERSITY OF MELBOURNE

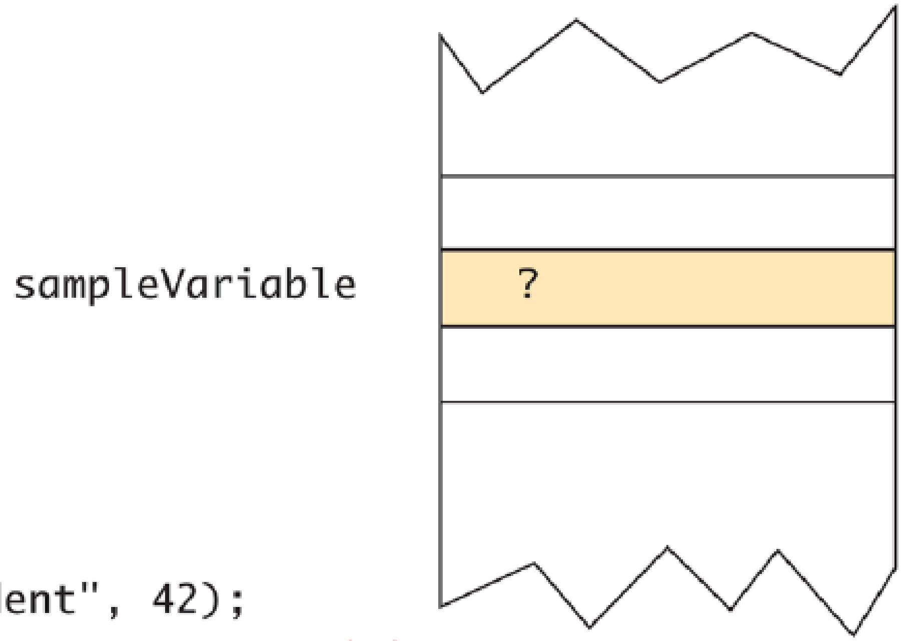
Display 5.12 Class Type Variables Store a Reference

```
public class ToyClass
{
    private String name;
    private int number;
```

The complete definition of the class `ToyClass` is given in Display 5.11.

```
ToyClass sampleVariable;
```

Creates the variable `sampleVariable` in memory but assigns it no value.

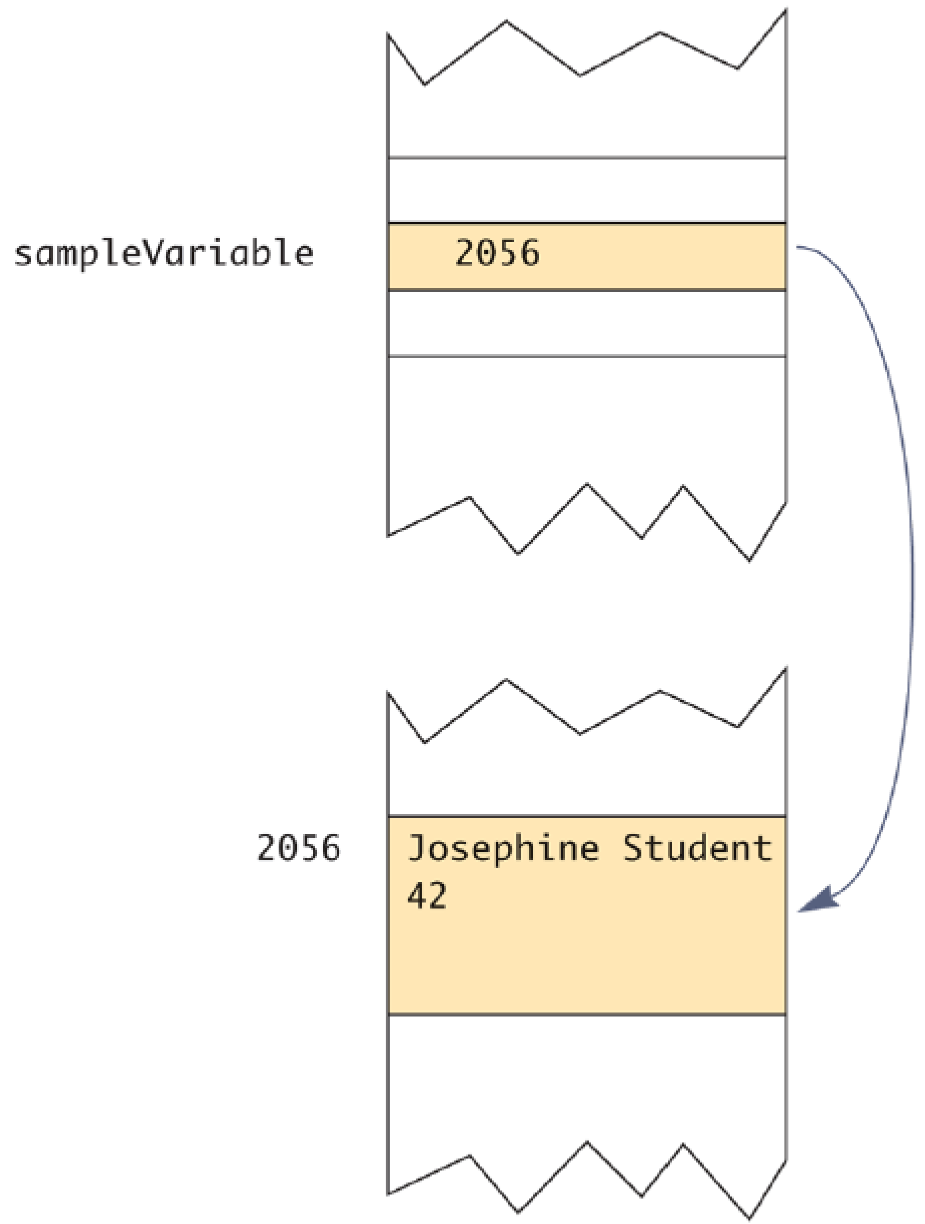


```
sampleVariable =
new ToyClass("Josephine Student", 42);
```

Creates an object, places the object someplace in memory, and then places the address of the object in the variable `sampleVariable`. We do not know what the address of the object is, but let's assume it is `2056`. The exact number does not matter.

(continued)

Display 5.12 Class Type Variables Store a Reference

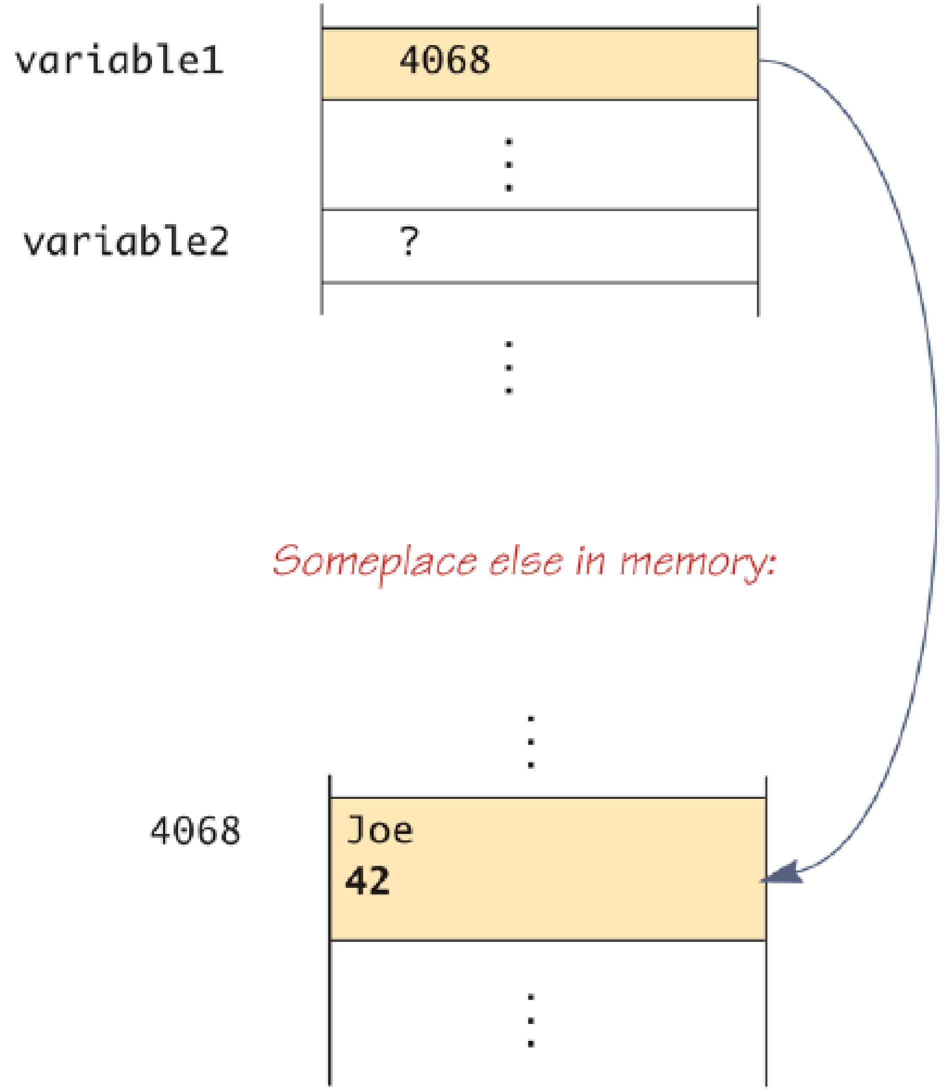


For emphasis, we made the arrow point to the memory location referenced.

THE UNIVERSITY OF MELBOURNE

Display 5.13 Assignment Operator with Class Type Variables

```
ToyClass variable1 = new ToyClass("Joe", 42);
ToyClass variable2;
```



We do not know what memory address (reference) is stored in the variable `variable1`*. Let's say it is* `4068`*. The exact number does not matter.*

Note that you can think of

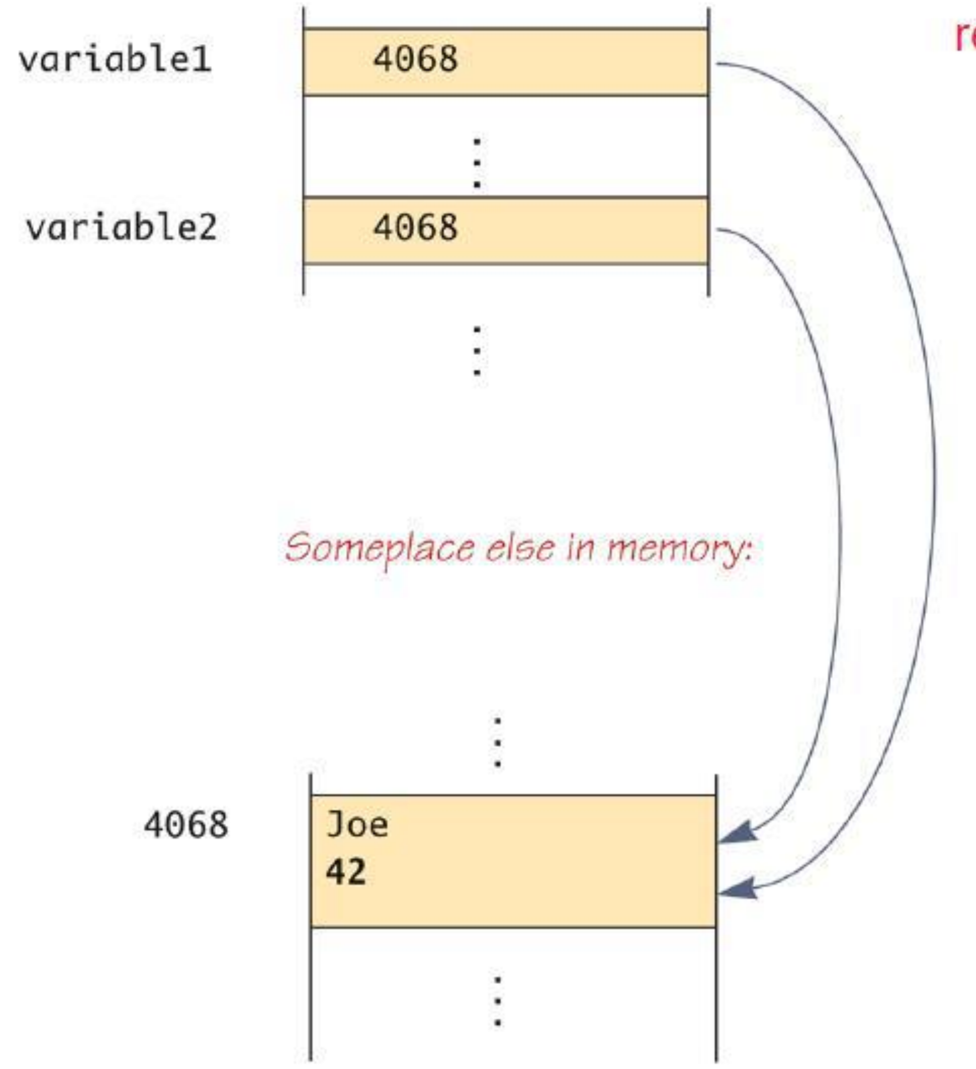
```
new ToyClass("Joe", 42)
```

as returning a reference.

(continued)

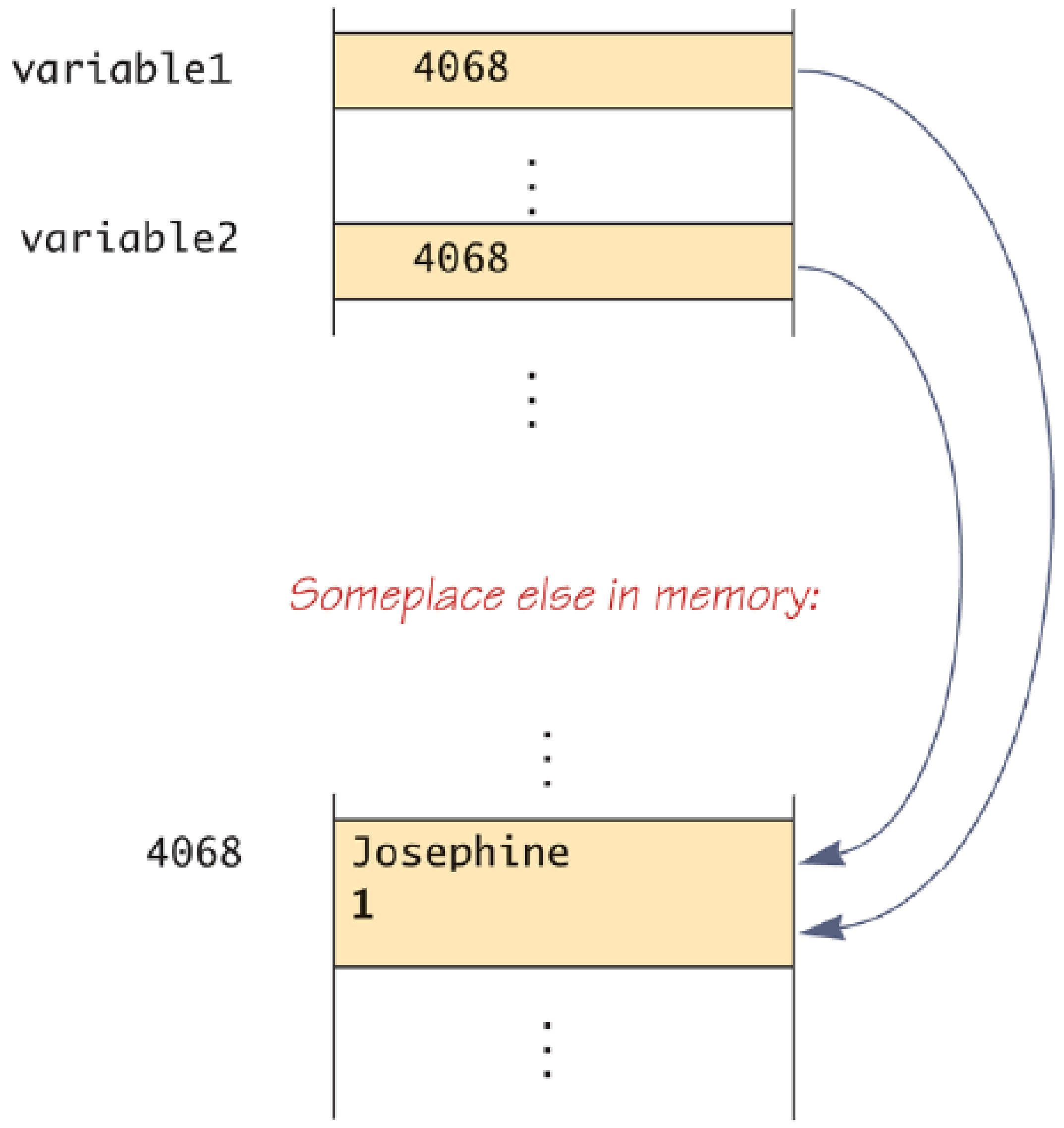
```
variable2 = variable1;
```

refer to the same memory address



(continued)

```
variable2.set("Josephine", 1);
```



the address not change
just the content

Java is Pass by Value

- ❖ Suppose I have an integer (primitive-type) variable num1, an instance of ToyClass toyObject, and a method myMethod
- ❖ int num1 = 10;
- ❖ ToyClass toyObject = new ToyClass();
- ❖ public void myMethod(int n1, ToyClass toy) {n1 = 20;}
- ❖ When I execute the method... myMethod(num1, toyObject);
- ❖ ...The value of num1 (i.e. 10) and the value of toyObject (i.e. the memory address 4068) will be passed to myMethod

Java is Pass by Value

- If we do not want to allow myMethod to change the original contents of our variable toyObject, we can pass a new variable with the copied contents of the object and whose value (memory address) will be different, e.g. using the copy constructor of ToyClass:
  - myMethod(num1, new ToyClass(toyObject);

Assignment 1

- Spec read-through
- Questions

THE UNIVERSITY OF MELBOURNE

Exercises