COMP 110-001
Objects and References

Yi Hong

June 01, 2015

Today

- Objects and references
- More on classes

Review

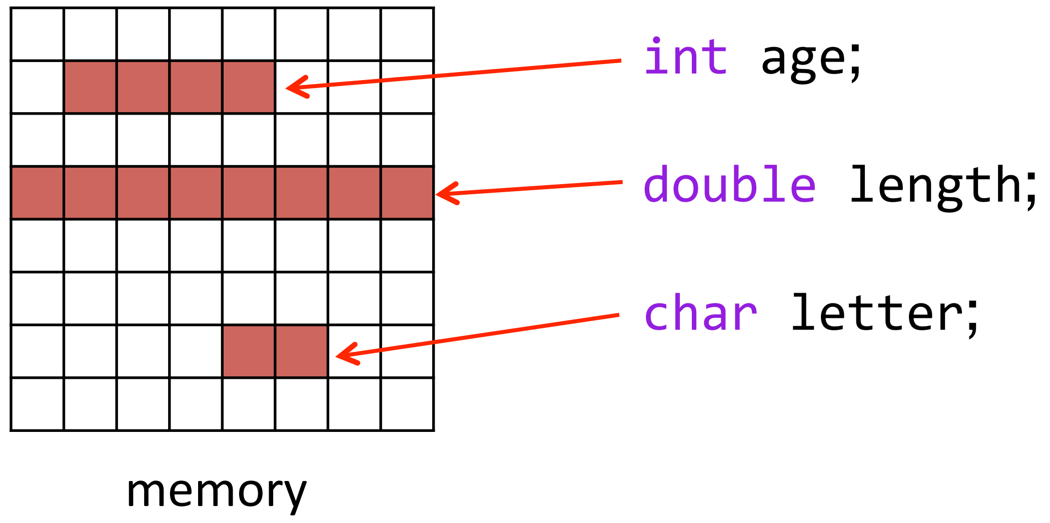
- Classes
- Objects
- Instance variables
- Methods
  - Return types
  - Parameters and arguments

Variables of a Class Type

- Behave differently from variables of a primitive type
  - Class types are reference types, a variable of a class type contains the memory address
  - A variable of a primitive type contains the data value

Variables of a Primitive Type

- When declaring a variable, a certain amount of memory is assigned based on the declared primitive type

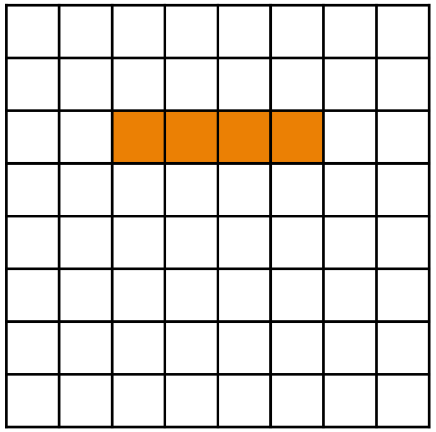


- What is in this memory?

Variables of a Primitive Type

- A data value is stored in the location assigned to a variable of a primitive type

```
int sum;

sum = 4;

sum = sum + 1;
```

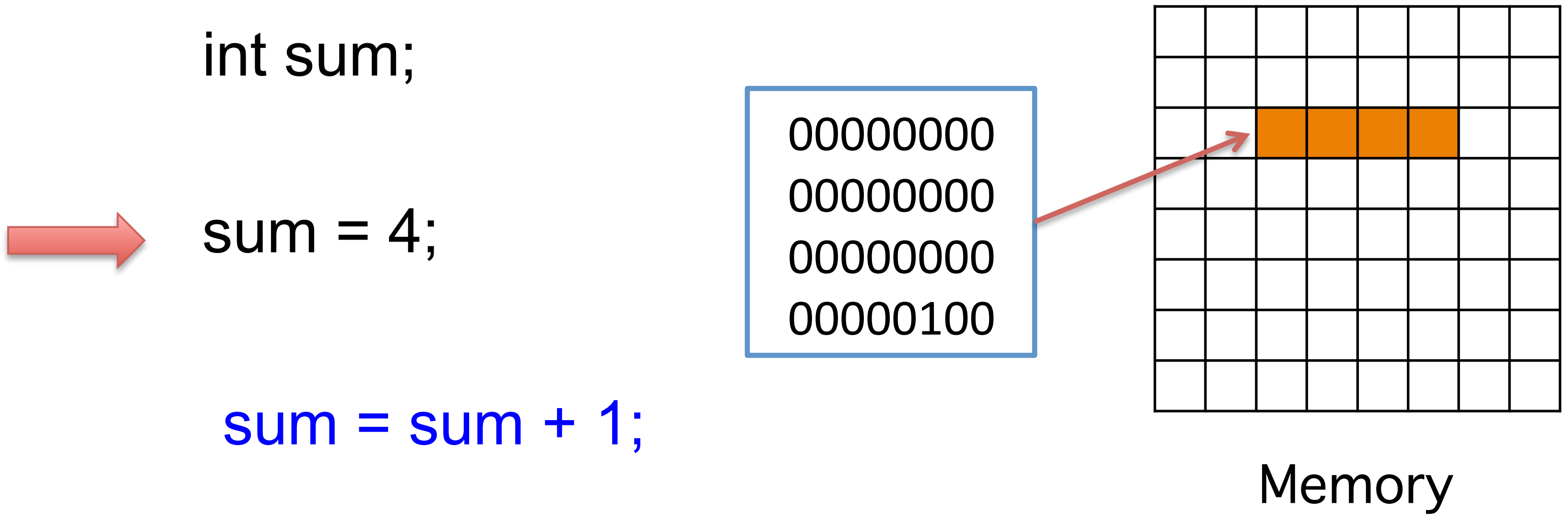


Memory

Variables of a Primitive Type

- A data value is stored in the location assigned to a variable of a primitive type

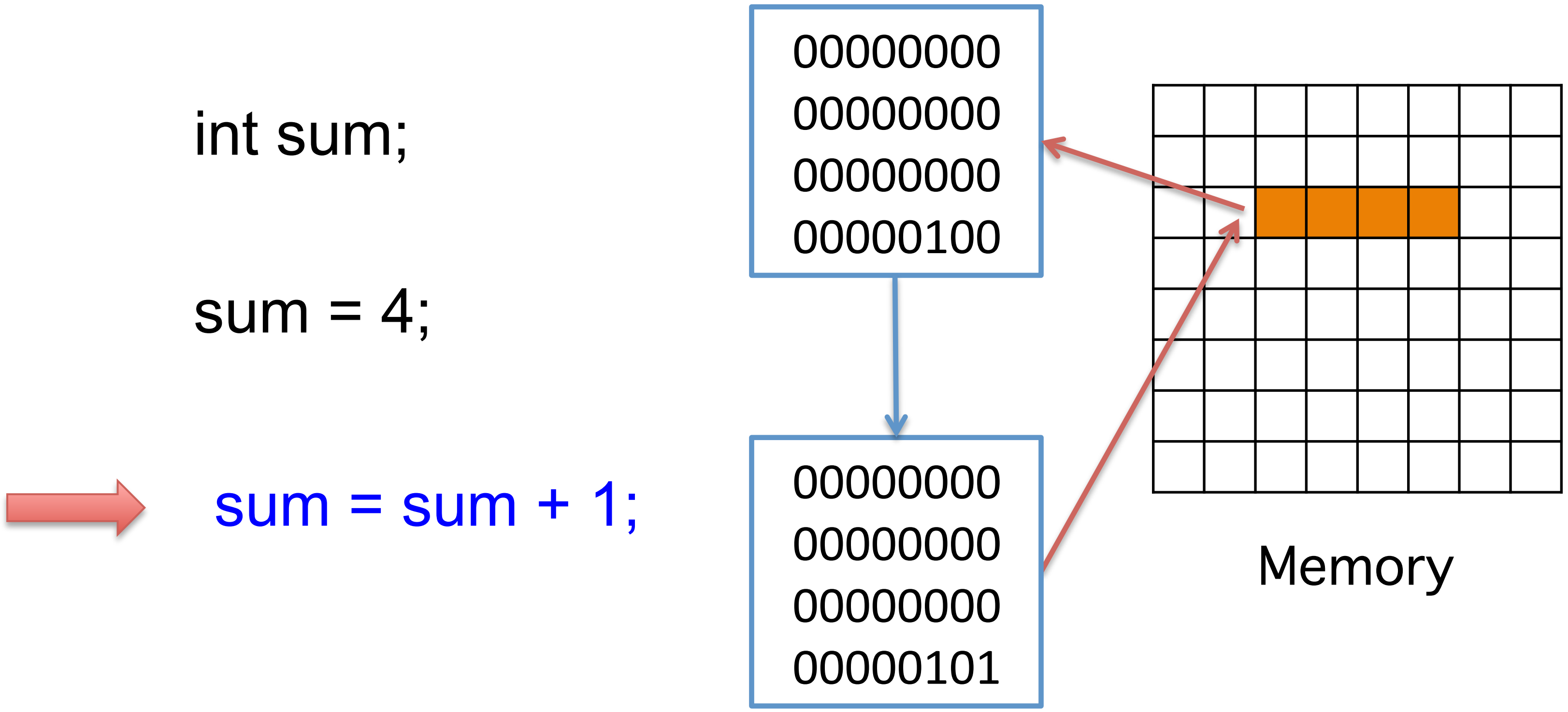
```
int sum;

sum = 4;

sum = sum + 1;
```



Variables of a Primitive Type

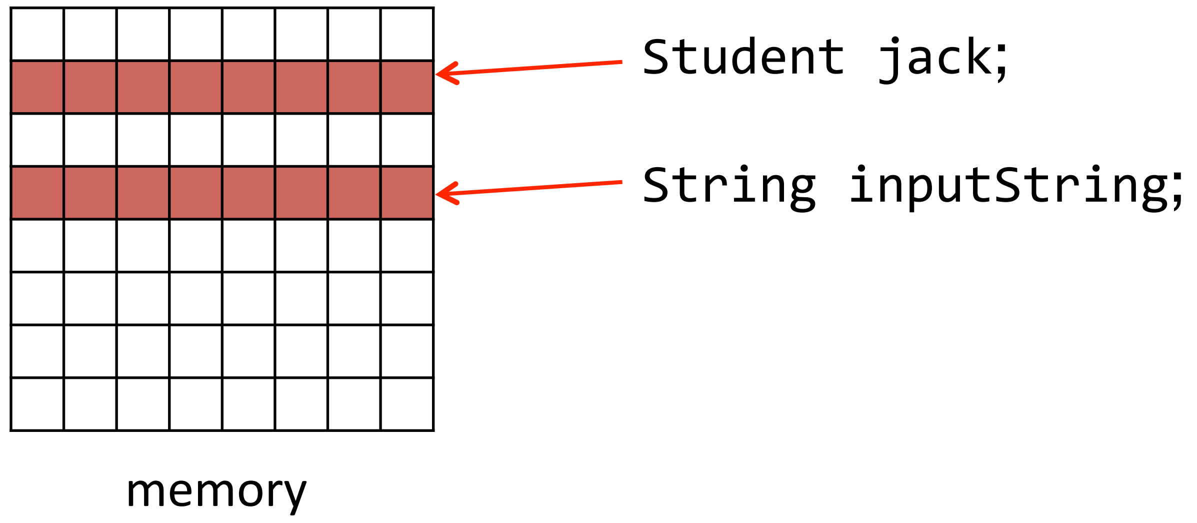
- A data value is stored in the location assigned to a variable of a primitive type

```
int sum;

sum = 4;

sum = sum + 1;
```



Variables of a Class Type

- What about these variables?



Variables of a Class Type

- Contain the memory *address* of the object named by the variable
  - NOT the object itself
- What is an address?
  - The object's location in the computer's memory
- Object is stored in some other location in memory
- The address to this other location is called a *reference* to the object
- Class types are also called *reference types*

Example: Books

Assume we have a class named Book

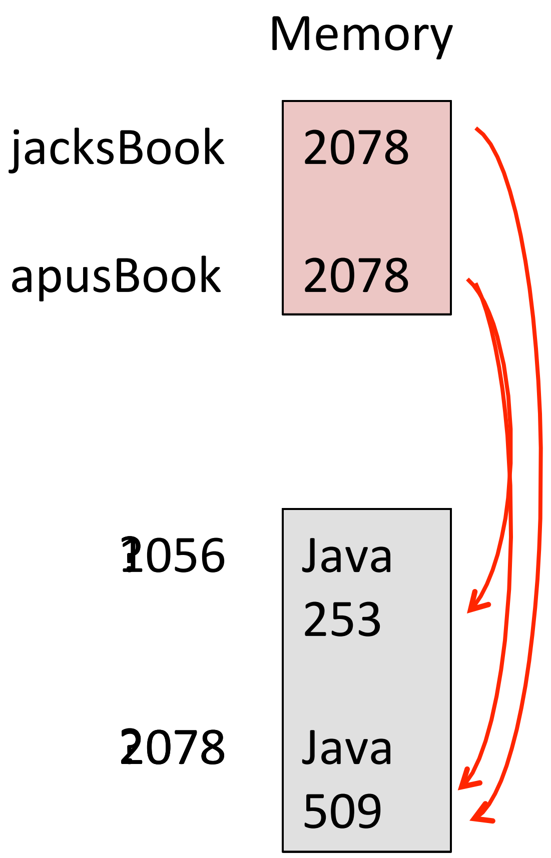
```
Book jacksBook = new Book();
Book apusBook = new Book();
```

vs.

```
Book jacksBook = new Book();
Book apusBook = jacksBook;
```

```
public class Book
{
    private name;
    private page;

    public void setName();
    public void setPage();
}
```

Objects in Memory



```
Book jacksBook;
Book apusBook;

jacksBook = new Book();
apusBook = new Book();

jacksBook.setName("Java");
apusBook.setName("Java");

jacksBook.setPage(137);
apusBook.setPage(253);

apusBook = jacksBook;
apusBook.setPage(509);
```

jacksBook is now on p. 509!

Remember

- Variables of a class type contain memory addresses
  - NOT objects themselves

== vs. equals() for Strings

- String is a class type
- What happens if you have

```
String s1 = new String( “Hello” );
String s2 = new String( “Hello” );
boolean strEqual = (s1 == s2);
```

- strEqual is false! Why?
- s1 and s2 store different addresses!

== vs. equals() for Strings

- What happens if you have

```
String s1 = new String( “Hello” );
String s2 = new String( “Hello” );
boolean strEqual = (s1.equals(s2));
```

- strEqual is true! Why?
- String’s .equals() method checks if all the characters in the two Strings are the same

Writing the .equals() method

```
public class Book
{
    private String name;
    private int page;

    public boolean equals(Book book)
    {
        return (this.name.equals(book.name) &&
                this.page == book.page);
    }
}
```

.equals()

- Every class has a default .equals() method if it is not explicitly written
    - Does not necessarily do what you want
- You decide what it means for two objects of a specific class type to be considered equal
    - Perhaps books are equal if the names and page numbers are equal
    - Perhaps only if the names are equal
    - Put this logic inside .equals() method

Call-by-value

- Java passes arguments to a method
- For primitive type, the parameter contains the value of its corresponding argument
- For class type, the reference (address) to the class object is passed to the parameters
  - Call-by-reference
  - It is possible to change the data in an object

Parameters of a Primitive Type

```
public void increaseNum(int num)
{
    num++;
}

public void doStuff()
{
    int x = 5;
    increaseNum(x);
    System.out.println(x);
}
```

Parameters are local to the method

- Prints 5. Why?
- num is local to increaseNum method; does not change x

Parameters of a Class Type

```
public void changeBook(Book book)
{
    book = new Book(“Biology”);
}

public void doStuff()
{
    Book jacksBook = new Book(“Java”);
    changeBook(jacksBook);
    System.out.println(jacksBook.getName());
}
```

Parameters are local to the method

- Prints Java. Why?
- book is local to changeBook, does not change jacksBook

Parameters of a Class Type

```
public void changeBook(Book book)
{
    book.setName(“Biology”);
}

public void doStuff()
{
    Book jacksBook = new Book(“Java”);
    changeBook(jacksBook);
    System.out.println(jacksBook.getName());
}
```

Parameters are local variables, but the reference is passed into the method

- Prints Biology. Why?
- book contains the same address as jacksBook!

Next Class

- Lab 6