Questions?
Today in COMP 110

- Objects and references
- More on Classes
Review

- Classes
- Objects
- Instance variables
- Methods
  - Return types
  - Parameters and arguments
- Information hiding and encapsulation
  - public/private
  - accessors/mutators
Variables of a class type

- Behave differently from variables of a primitive type
Variables of a primitive type

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

```
int age;
double length;
char letter;
```

- What goes in this memory?
Variables of a primitive type

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

- What goes in these variables?

```
Student jack;
String inputString;
```

memory
Variables of a class type

- Contain the memory address of the object named by the variable
  - NOT the object itself
- What is an address?
- 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
Assume we have a class named Book

```java
Book jacksBook = new Book("Java");
Book apusBook = new Book("Java");

VS.

Book jacksBook = new Book("Java");
Book apusBook = jacksBook;
```
Objects in memory

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
## Objects in memory

<table>
<thead>
<tr>
<th>Memory</th>
<th>Book jacksBook; Book apusBook;</th>
</tr>
</thead>
<tbody>
<tr>
<td>jacksBook</td>
<td>?</td>
</tr>
<tr>
<td>apusBook</td>
<td>?</td>
</tr>
</tbody>
</table>
| ... | ...
| ... | ...
| ... | ...
| ... | ...
Objects in memory

```java
Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
```

Memory

<table>
<thead>
<tr>
<th>jacksBook</th>
<th>1056</th>
</tr>
</thead>
<tbody>
<tr>
<td>apusBook</td>
<td></td>
</tr>
<tr>
<td>&quot;Java Book&quot;</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Book object</td>
<td>1056</td>
</tr>
</tbody>
</table>

Objects in memory

Book object

Book object

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>jacksBook</td>
</tr>
<tr>
<td>apusBook</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>“Java Book”</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>“C++ Book”</td>
</tr>
<tr>
<td>?</td>
</tr>
</tbody>
</table>

Book jacksBook;
Book apusBook;

jacksBook = new Book(“Java”);
apusBook = new Book(“C++ Book”);
Objects in memory

Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
apusBook = new Book("C++ Book");
jacksBook.setPage(68);
# Objects in memory

```java
Book jacksBook;  
Book apusBook;

jacksBook = new Book("Java");  
apusBook = new Book("C++ Book");

jacksBook setPage(68);  
apusBook setPage(254);
```

<table>
<thead>
<tr>
<th>Memory</th>
<th>jacksBook</th>
<th>Book object</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1056</td>
<td>“Java Book”</td>
<td>1056</td>
</tr>
<tr>
<td></td>
<td>2078</td>
<td>68</td>
<td>2078</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>“C++ Book”</td>
<td>254</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Objects in memory

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>jacksBook</td>
</tr>
<tr>
<td>apusBook</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>“Java Book”</td>
</tr>
<tr>
<td>jacksBook</td>
</tr>
<tr>
<td>apusBook</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>“C++ Book”</td>
</tr>
</tbody>
</table>

Book objects

```java
Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
apusBook = new Book("C++ Book");
jacksBook.setPage(68);
apusBook.setPage(254);
apusBook = jacksBook;
```
Objects in memory

Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
apusBook = new Book("C++ Book");
jacksBook setPage(68);
apusBook setPage(254);
apusBook = jacksBook;
apusBook setPage(316);
Objects in memory

Book object

jacksBook

apusBook

Book object

"Java Book"
316

"C++ Book"
254

Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
apusBook = new Book("C++ Book");

jacksBook.setPage(68);
apusBook.setPage(254);

apusBook = jacksBook;
apusBook.setPage(316);

jacksBook is now on p. 316!
Remember

- Variables of a class type contain memory addresses
  - NOT objects themselves
String s1 = new String("Hello");
String s2 = new String("Hello");
boolean strEqual = (s1 == s2);

strEqual is false! Why?
s1 and s2 store different addresses!
What happens when 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
public class Book {
    private String name;
    private int page;

    public boolean equals(Book book) {
        return (this.name.equals(book.name) &&
                this.page == book.page);
    }
}
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
Parameters of a primitive type

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

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

- Prints 5. Why?
Parameters of a primitive type

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

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

- Prints 5. Why?
  - num is local to increaseNum method
    - does not change x
public void changeBook(Book book)
{
    book = new Book("Biology");
}

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

 Prints Java. Why?
public void changeBook(Book book)
{
    book = new Book("Biology");
}

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

- Prints Java. Why?
- book is local to changeBook, does not change jacksBook
public void changeBook(Book book)
{
    book.setName("Biology");
}

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

- Prints Biology. Why?
Parameters of a class type

```java
public void changeBook(Book book) {
    book.setName("Biology");
}

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

- Prints Biology. Why?
- book contains the same address as jacksBook!
Write an inventory class for pet store

It needs to have the ability to track

- # of frogs
- # of birds
- # of turtles
- 3 transaction types
  - Add
  - Remove
  - Inquiry
Would you like to make a transaction? y/n
y
Would you like to make a add (a), remove (r), or inquire (i)?
a
How many frogs, birds, and turtles would you like to add?
Frogs: 7
Birds: 6
Turtles: 4
Would you like to make a transaction? y/n
y
Would you like to make a add (a), remove (r), or inquire (i)?
r
How many frogs, birds, and turtles would you like to remove?
Frogs: 4
Birds: 0
Turtles: 0
Would you like to make a transaction? y/n
y
Would you like to make a add (a), remove (r), or inquire (i)?
i
You now have:
3 frogs
6 birds
4 turtles
public class PetInventory {
    private int frogs = 0;
    private int birds = 0;
    private int turtles = 0;
}
Accessor and Mutator Methods

- getFrogs: int
- getBirds: int
- getTurtles: int
- setFrogs(int)
- setBirds(int)
- setTurtles(int)
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.add(5, 7, 9);
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.add(5, 7, 9);
}

public void add(int f, int b, int t) {
    frogs += f;
    birds += b;
    turtles += t;
}
Add

```java
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.add(5, 7, 9);
}
```

```java
public void add(int f, int b, int t) {
    frogs += f;
    birds += b;
    turtles += t;
}
```
```java
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.add(5, 7, 9);
}

public void add(int f, int b, int t) {
    frogs += f;
    birds += b;
    turtles += t;
}
```
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.remove(5, 7, 9);
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    ...
    petInventory.remove(5, 7, 9);
}

public void remove(int f, int b, int t) {
    if ((f > frogs) || (b > birds) || (t > turtles)) {
        System.out.println("You do not have enough pets for that!");
    } else {
        frogs -= f;
        birds -= b;
        turtles -= t;
    }
}
Inquire

```java
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    .
    .
    .
    petInventory.inquire();
}
```
public static void main(String[] args) {
    PetInventory petInventory = new PetInventory();
    petInventory.inquire();
}

public void inquire() {
    System.out.println("You now have: " + frogs + " frogs, " + birds + " birds, and " + turtles + " turtles.");
}
Tomorrow

- Midterm Review
- Bring questions about the Sample Midterm