Review for the Third Exam

GEEN163

“I was thinking about how people seem to read the Bible a whole lot more as they get older; then it dawned on me - they're cramming for their final exam.”

George Carlin

Remaining Schedule

	Wednesday, April 26 review	Friday, April 28 **Exam 3**
Monday, May 1 Software engineering	Monday, May 3 final review lab	final
	Wednesday, May 10 8:00am – 10:00am **Final Exam**	

Exam 3 Topics

book section

- classes 7
  - constructors
  - class variables
    - static and non-static class variables
  - creating objects
- methods 7
  - parameter passing
- scoping 7
- arrays 8

Anything is fair game

- The exam may contain questions from any of the material covered in class since the last exam
- Knowledge is cumulative, so using the newer topics may require you to know previous topics

One Page of Notes

- You are allowed one and only one 8½ by 11 inch page of notes during this exam
- You are not allowed to use more than 187 square inches of paper surface
- You will do better if you make your own page of notes and not copy your friend’s notes

Practice Exam

- A practice exam from a previous semester is on Blackboard under course materials

Exam Format

- The exam will be similar to previous exams, labs, quizzes and homework
- Exams tend to be programming oriented
- Most question will be *“write a method to”*, *“declare an array of”* or *“what does this program display”*

Secure Programming

- A good program checks all input values to make sure they are reasonable
- Numbers should be within range
- This not only makes the program more secure, but it improves the user interface
- Be aware of the possibility of integer overflow
- Avoid characters that may signal code injection

Write with your team

- Write a method that returns `true` if the string goat contains only letters and spaces. Return `false` if it contains other characters

```
boolean isSafe(String goat)
```

- You can use the static method boolean isLetter(char ch) of the Character class and the charAt(int pos) method of String

Possible Solution

```
boolean isSafe(String goat){
    for (int i = 0; i < goat.length; i++) {
        char kid = goat.charAt( i );
        if (!Character.isLetter(kid)
                && letter != "") {
            return false;
        }
    }
    return true;
}
```

Write a Constructor

- Write a constructor to initialize bass and trout to parameters

```
public class Fish {
        private int bass;
        private double trout;
}
```

Possible Solution

```
public class Fish {
        private int bass;
        private double trout;
        public Fish( int carp, double pike ) {
                bass = carp;
                trout = pike;
        }
}
```

Which method will get the bass value?

A. `void setBass( int shark ) { bass = shark; }`
B. `int getBass() { return bass; }`
C. `void getBass( int shark ) { return bass; }`
D. `int getBass() { bass = shark; }`
E. none of the above

Write an equals method for Fish

- Write the method

  boolean equals(Fish other)

- that returns true if the bass and trout values of the other object are the same as the this object's bass and trout
- This method will **override** the equals method of Object

Possible Solution

```
public class Fish {
    private int bass;
    private double trout;
    boolean equals( Fish other ) {
        return bass == other.bass  &&  trout == other.trout ;
    }
}
```

Possible Solution

```
public class Fish {
        private int bass;
        private double trout;
        boolean equals( Fish other ) {
         return other != null  &&       // check for null
                bass == other.bass  &&  trout == other.trout ;
        }
}
```

Use the Fish Class

- Create an object of the Fish class setting trout to 4.25 and bass to 163

Possible Solution

- Create an object of the Fish class setting trout to 4.25 and bass to 163

```
Fish minnow = new Fish( 163, 4.25);
```

Initializing Arrays

- Like simple variables, arrays can be initialized to a value when they are declared
- When you initialize an array, you do not need to specify the size. The size is determined by how many values you use to initialize

```
int[] rabbit = { 32, 14, 7, 26, 86};
```

- The initialization values must be the correct type
- Note the semicolon after the curly bracket

Arrays of Strings

- An array of Strings can be initialized

```
String[] holiday = { “Christmas”,
                     “Thanksgiving”, “Independence Day” };
```

- With any array initialization, the data values must be separated by commas

Loops

- When you are dealing with the elements of an array, you almost always use a loop

```
//  Add together all values of the array bear
double[] bear = new double[5000];
double sum= 0.0
for (int deer = 0; deer < bear.length; deer++) {
        sum += bear[ deer ];
}
```

Arrays as Objects

- An array is an object in Java
- You can call a method on the array
- Arrays have an integer class variable length that contains the length of the array

```
double[] cobra = new double[10000];
int boa = cobra.length;                    // boa = 10000
```

Half Empty or Half Full

- An array might not contain as many useful values as its full capacity

```
Widget[] eagle = new Widget[100];
for (int egg = 0; egg < 47; egg++) {
        eagle[egg] = new Widget();
}
```

- The array eagle has 100 elements, but only 47 values have been used
- eagle[60].getCount() will cause a nullPointerException

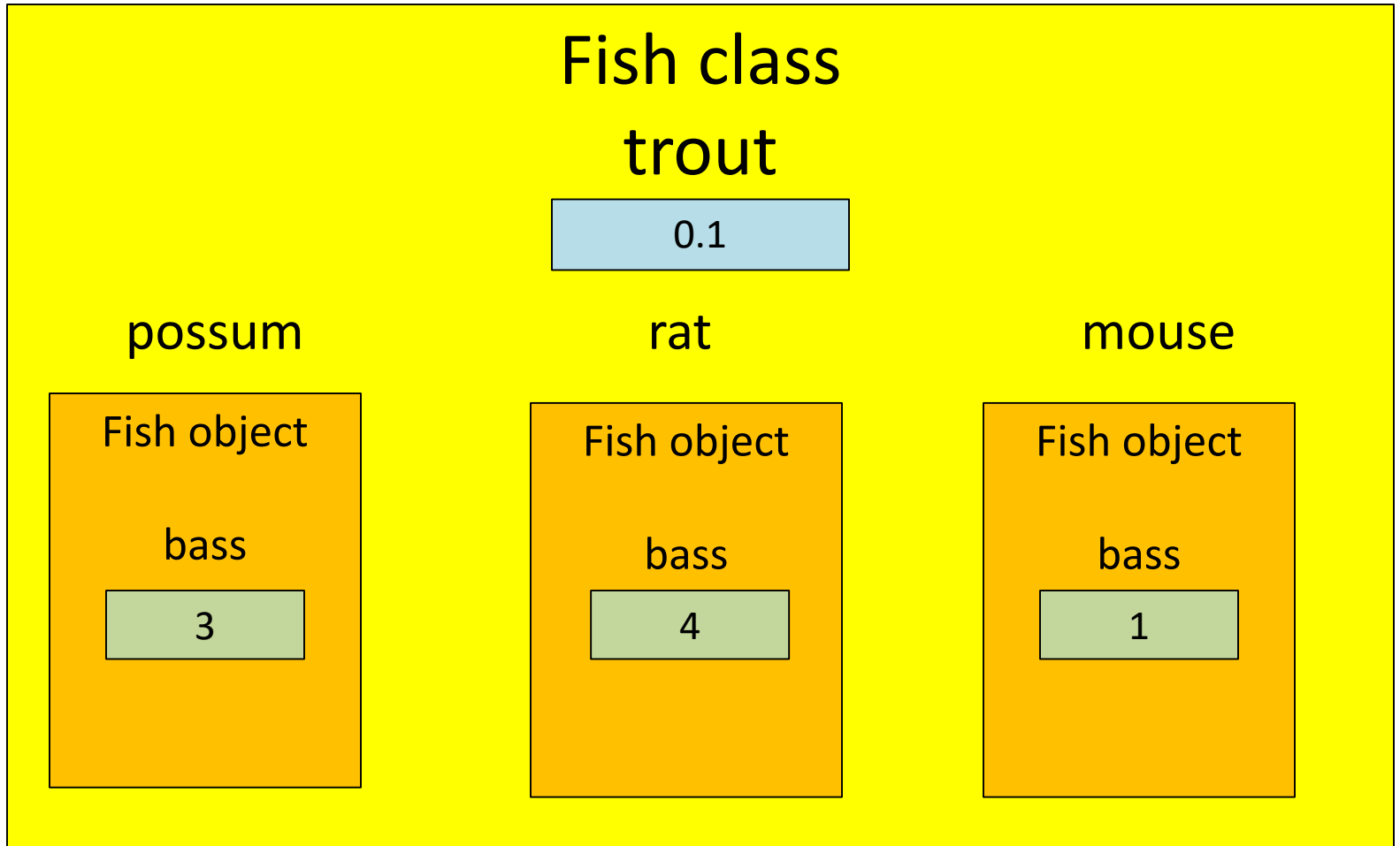
Multiple Objects

```
public class Fish {
    private int bass;
    private static double trout;
}
Fish possum = new Fish(3, 5.5 );
Fish rat    = new Fish( 4, 8.8 );
Fish mouse  = new Fish( 1, 0.1);
```

- There are three Fish objects
- Each object has its own copy of bas
- There is only one trout variable

Static and Instance Variables

- There is only one trout variable, but there is a dog for every object



What is displayed?

```
public class Stest {
   static int svar = 1;
   int dvar = 1;
   void incBoth() {
      svar++;
      dvar++;
      System.out.println(svar+" "+dvar);
   }
   static public void main(String[] x) {
      Stest cat = new Stest();
      Stest dog = new Stest();
      cat.incBoth();
      dog.incBoth();
   }
}
```

A. 1 1
   1 1

B. 2 2
   2 2

C. 2 2
   3 3

D. 2 2
   3 2

What is printed by this program?

```
public class Click {
 static int      crow= 5, raven= 7;
 static void myfunc(int parrot) {
       int crow;
       crow = parrot + 1;
       System.out.print( crow );
 }
 public static void main(...) {
    Click ques = new Click();
    ques.myfunc( 3 );
    System.out.print( crow );
  }
}
```

A. 4 4

B. 4 5

C. 5 5

D. none of above

Creating Objects from Files

- A program might declare an array of objects in the beginning and then create the object later
- Consider a program that reads a data file and creates an object for each line of data

Complete the program

- Create an array of Fish objects from the data in lake.txt which contains two numbers per line

```
java.io.File pond = new java.io.File( “lake.txt” );
java.util.Scanner inFile = new java.util.Scanner( pond );
Fish[] lake = new Fish[128];
int numFish = 0;
while (inFile.hasNextInt()) {
```

Possible Solution

- Create an array of Fish objects from the data in lake.txt which contains two numbers per line

```
java.io.File pond = new java.io.File( “lake.txt” );
java.util.Scanner inFile = new java.util.Scanner( pond );
Fish[] lake = new Fish[128];
int numFish = 0;
while (inFile.hasNextInt()) {
        int guppie = inFile.nextInt();
        double cod = inFile.nextDouble();
        lake[numFish] = new Fish( guppie, cod );
        numFish++;
}
```

Write with your Neighbors

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog )
```

Possible Solution

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog ){
    int count = 0;
    for (double cat : dog) {
        if (cat < 0) {
            count++;
        }
    }
    return count;
}
```

Another Possible Solution

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog ){
    int count = 0;
    for (int cat=0; cat <dog; cat++) {
        if (dog[cat] < 0) {
            count++;
        }
    }
    return count;
}
```

Putting a Positive Spin on It

- Consider a modification to our method that makes all of the negative number a positive number of the same magnitude

Will this work?

- Make all the numbers positive

```
int  makePositive( double[] dog ){
    int count = 0;
    for (int cat=0; cat <dog; cat++) {
        if (dog[cat] < 0) {
            dog[cat] = -dog[cat];
        }
    }
    return count;
}
```

A. No – you cannot change the array
B. No – you can't put a dash before dog
C. All the above
D. Yes

Looping in GUI Programs

- When you want a user to enter multiple values in a GUI program, you probably do **not** need a for loop or a while loop
- Each time the user enters a name, it will call actionPerformed

Making a GUI Application

1. Create a class that extends JFrame and implements java.awt.event.ActionListener
2. Create component objects
3. Specify the layout manager in the constructor
4. Add the component objects to the container
5. Add an action listener to the components that respond to user input
6. Create an actionPerformed method

Likely Questions

- Create and initialize an array
- Write a class with constructor and other methods
- Evil questions with class and local variables that have the same name
- Write an actionPerformed method

Remaining Schedule

	Wednesday, April 26 review	Friday, April 28 **Exam 3**
Monday, May 1 Software engineering	Monday, May 3 final review lab	final
	Wednesday, May 10 8:00am – 10:00am **Final Exam**	