Please read Sections 6.6 through 6.10 and Section 6.12 of the textbook and then answer the following, trying not to look at your notes or at the textbook. Quiz #8, on Fri. 28 Oct., will consist exclusively of questions taken from the Part 1 of this homework.

Ex. 1. What is wrong with the following code?

```
public class MyClass{
    private int x;
    public static void setValues(int a){
        x = a;
    }
}
```

Ex. 2. What happens when an object is passed as an argument to `print` or `println`?

Ex. 3. Do all classes have a `toString` method?

Ex. 4. Why do we need to write `equals` methods to compare two objects from the same class?

Ex. 5. Assuming the `Rectangle` class has being defined, and that it has a constructor taking two arguments, what would the following code print? Can you explain why?

```
Rectangle box1 = new Rectangle(1, 20);
Rectangle box2 = new Rectangle(1, 20);
if (box1 == box2)
    {System.out.print("A");}
else
    {System.out.print("B");}
```

Ex. 6. Write a public method named `rectangleCopy` that copy a `Rectangle` object (remember that a `Rectangle` has two fields of type `int`, named `length` and `width`).

Ex. 7. What is the name of the reference variable that is always available to an instance method and refers to the object that is calling the method?

Ex. 8. What is a reference copy?

Ex. 9. Would the following code compile? If you think it would, tell what would be printed when executed, otherwise tell the sort of error we'll meet.

```
String myName;
System.out.print(myName.length());
```

Ex. 10. Write an enumerated data type declaration that creates an enumerated data type named `Section`, with constants `CS102`, `CS103`, `CS104`. In your declaration, what is the ordinal value of the constant `CS102`?

Ex. 11. Assume the `enum` constant `APRIL` belongs to the `Month` enumerated data type. Write a statement that declares a variable whose type is this enumerated data type, and initialize it with `APRIL`.

Ex. 12. Suppose we an `enum` data type `Feline` containing the constant `JAGUAR` and `LION`. What will the following statements display?

```
System.out.println(Feline.JAGUAR + "\n" + Feline.LION);
```

Ex. 13. Assume we have the following enumerated types: `enum Rooms{BH310, BH311, BH312F}`. Would the following test evaluates to `true` or `false`? Explain why.

```
Rooms.BH310.compareTo(Rooms.BH311) > 0
```

Part II — Programming Exercises

Warning: this week's programming exercise is challenging. But there's only one exercise, so take your time!

Ex. 1. We're going to construct a small program that converts dollars ($) in euros (€). We'll need three files: `Currency.java`, that contains the declaration of an `enum` data type, `Money.java`, that contains the definition of a `Money` class, and an application program, `MoneyDemo.java`.

`Currency.java` will contain only the following:

```
enum Currency{DOLLAR, EURO}
```

`Money.java` should implement the following UML diagram:

Money
- currency: Currency - value: double
+ Money() + Money(currency: Currency, value: double) + getCurrency(): Currency + getValue(): double + convert(): Money + toString() : String

The definition of the two fields shouldn't be too difficult, and the `no-arg` constructor should do nothing. The other constructor should do the obvious thing of initializing the value of the `currency` field with the first parameter, and the value of the `value` field with the second parameter. Notice that you should use the `this` keyword. The two getters behave as usual.

Next, we want to define a `convert` method that creates a `money` object that corresponds to the conversion in the other currency of the calling object. Decide what the exchange rate should be, or find it on he Internet. Finally, the `toString` method should just return a string made of either "$" and then the `value` field, or the `value` field followed by "€". In both cases, you should display the `value` with only two decimal places.

`MoneyDemo.java` should do the following:

- Ask the user to chose between dollars and euros. We should ask the user as long as (s)he did not entered a valid choice.
- Ask the user an amount of dollar, or of euro, depending on the previous choice.
- Create a `Money` object with the data given by the user.
- Call the `convert` method with the object we just created to create a `money` object that corresponds to the conversion.

Listing 1: A First Example

```
Enter D for dollars, E for euros:
P↵
Enter D for dollars, E for euros:
E↵
Enter an amount of euros:
90↵
90.00€ is $98.36
```

Listing 2: A Second Example

```
Enter D for dollars, E for euros:
D↵
Enter an amount of dollars:
192↵
$192.00 is 175.68€
```

- Display, using the `toString` method, both objects.

Think about your program, try to get as much done as possible before turning on a computer. Then write your three programs in the order listed, and make sure `Currency.java` compiles before working on `Money.java`.