Object Oriented Design

Constructors

A class contains constructors that are invoked to create objects from the class blueprint. Constructor declarations look like method declarations—except that they use the name of the class and have no return type.For example, Bicycle has one constructor:

```
public Bicycle(int startCadence, int startSpeed, int startGear) {
        gear = startGear;
        cadence = startCadence;
        speed = startSpeed;
}
```

To create a new Bicycle object called myBike, a constructor is called by the new operator:

```
Bicycle myBike = new Bicycle(30, 0, 8);
```

new Bicycle(30, 0, 8) creates space in memory for the object and initializesits fields.

(Exerpt taken from java.sun.com)

Provide multiple constructors in any of your classes and demonstrate how it works

(Credit: 4 Pts)

Super Constructors (2 Pts)

When a class extends another class it is possible to call the parent class constructor. This is done by using the keyword super() inside the child's constructor.

Provide a good example in your code of such a use for the super keyword.

Credit (2 Pts)

This keyword (2 Pts)

`this`

is a keyword in java that allows you to reference the current object.

This is especially useful in accessing variables.

Change your accessor methods (get and set) in one of your programs to effectively use the this keyword.

Example

```
public void setValue (int val) {
```

```
    this.val = val;   //you need to see that both variables have the same name but are different because of the this keyword

}
```

Effective Coding Design (8 Pts)

Take one of your larger programs and provide the appropriate level of visibility for methods and functions

public - means the method or variable is visible outside the class
private - means the method or variable is visible only within the class
protected - means the method or variable is visible only within the class or any class that extends the current class.

You should also be able to answer the following questions

1) What happens if a function outside the class tries to access a variable declared as private in the class

2) What happens if a function outside the class tries to access a variable declared as protected in the class

3) What is the visibility of a method or variable that is not declared as public private or protected

Final Keyword (3 Pts)

The final keyword serves two purposes.

First for data it makes the variable immutable (unchangeable), this makes the variable a constant. This enables the compiler to replace at compile time the variable with the actual constant making execution faster.

Second, a method that is final locks the method, to prevent any inherited classes from changing the method. This ensures that the method is never overridden. In addition, the compiler will take final methods and insert their code directly into the body of the code where it would be called increasing the efficiency.

Your assignment is to create simple constants for the following PI, E, NAUTICAL_MILE, and the GOLDEN_RATIO.

Second, explain why the main method is always created as final.

Credit (3 Pts)

Static Keyword

The static keyword is used in two cases (methods and variables), similar to the final keyword.

A variable that is declared as static doesnt need an instance created in order to access it. Object variables have different copies for each instance. A class variable has only a single copy of a variable share for the class.

It becomes a "Class Variable" as opposed to an object Variable.

For example

```
class Sample {
```

```
    static int x = 6

}


...
```

can be called as follows

Sample.x

However if the static keyword were no in front of the variable, you would need to call it as follows

```
Sample s = new Sample()
s.x   <- notice the use of the object not the class name - hence the differnce between a class variable and an object variable.
```

---- Static Methods

The use of static methods is similar in the sense that you do not need to declare an instance of an object to call the variable.
Simply call the method using the class name,

Example:

```
Sample.addnumber(2);
```