Constants

Use keyword `final`

```
final double MY_CONST .7894;

private static final MY_CONST .784;


public static final MY_CONST .784;

double circumference = Math.PI * diameter;
```

Javadoc utility

- Creates HTML documentation
- Comments of special form

```
/**

*/
```

- First sentence of each comment goes to summary table. It should begin with a capital letter and end with a period.
- A comment is supplied for each method and class
- Special keywords

```
@param
@return
```

- `javadoc MyClass.java`

Example

```
/**
   A purse computes the total value
    of a collection of coins.
*/
public class Purse
{
   /**
      Constructs an empty purse.
   */
   public Purse()
   {
      nickels = 0;
      dimes = 0;
      quarters = 0;
   }
```

Example (cont)

```
/**
   Add nickels to the purse.
   @param count the
     number of nickels to add
*/

public void addNickels(int count)
{
   nickels = nickels + count;
}
```

Example (cont)

```
/**
   Add dimes to the purse.
   @param count
    the number of dimes to add
*/
public void addDimes(int count)
{
   dimes = dimes + count;
}
```

Example (cont)

```
/**
   Add quarters to the purse.
   @param count the number
    of quarters to add
*/
public void addQuarters(int count)
{
   quarters = quarters + count;
}
```

Example Continued

```
/**
   Get the total value of
       the coins in the purse.
   @return the sum of all coin values
*/
public double getTotal()
{
   return nickels * NICKEL_VALUE
      + dimes * DIME_VALUE
         + quarters * QUARTER_VALUE;
}
```

Example Continued

```
    private static final double
                        NICKEL_VALUE = 0.05;
    private static final double
                        DIME_VALUE = 0.1;
    private static final double
                        QUARTER_VALUE = 0.25;
    private int nickels;
    private int dimes;
    private int quarters;
}
```

Reading Input

JOptionPane from javax.swing

```
String input =
 JOptionPane.showInputDialog("Give number");

int count = Integer.pareseInt(input);
System.exit(0);
```

Example

```
import javax.swing.JOptionPane;
public class InputTest
{
  public static void main(String[] args)
  {
  Purse myPurse = new Purse();
  String input JOptionPane.showInputDialog(
    "How many nickels do you have?");
  int count = Integer.parseInt(input);
  myPurse.addNickels(count);
  String input JOptionPane.showInputDialog(
    "How many dimes do you have?");
  count = Integer.parseInt(input);
  myPurse.addDimes(count);
```

Example

```
String input JOptionPane.showInputDialog(
  "How many quarters do you have?");
count = Integer.parseInt(input);
myPurse.addQuarters(count);
double totalValue = myPurse.getTotal();
System.out.println("The total is " + totalValue);
System.exit(0);
```

Type Conversion

```
double total;
int pennies = (int) total;

double price = 44.95;

int dollars = (int) (price + .5);

int dollars = (int)Math.round(100*f);
```

Static Initialization

```
class Primes {
   static int[] knownPrimes = new int[4];

   static {
      knownPrimes[0] = 2;
      for (int i = 1; i <
            knownPrimes.length; i++)
          knownPrimes[i] = nextPrime();
      }
   // declaration of nextPrime
}
```

Methods

```
class BodyPrint {
  public static void main(String[] args) {
    Body sun = new Body("Sol", null);
    Body earth = new Body("Earth", sun);
    System.out.println("Body " +
                                earth.name +
                                "orbits "+
                                earth.orbits.name +
                              "and has ID " +
                                earth.idNum);
```

Method Invocations

reference.method(arguments)

```
public String toString() {
    String desc = idNum + " (" + name + ")";
    if (orbits != null)
      desc += " orbits " + orbits.toString();
    return desc;
   }
```

Method Execution and Return

```
public class Permissions {
   public boolean canDeposit,
                  canWithdraw,
                  canClose;
  }
```

Method Execution and Return (cont)

```
public class BankAccount {
    private long number; //account number
    private long balance; //current balance
    public Permissions
            permissionsFor(Person who) {
      Permissions perm = new Permissions();
      perm.canDeposit = canDeposit(who);
      perm.canWithdraw = canWithdraw(who);
      return perm;
     }
// .. define canDeposit et al...
}
```

Parameter Values

```
class PassByValue {
   public static void main(String[] args){
     double one = 1.0;
     System.out.println("before: one = " + one);
     halveIt(one);
     System.out.println("after: one = " + one);
   }

   public static void halveIt(double arg) {
     arg /= 2.0; // divide arg by two
     System.out.println("halved: arg = " + arg);
   }
}
```

Parameter Vals (cont)

```
class PassRef {
   public static void main(String[] args){
     Body sirius = new Body("Sirius", null);
     System.out.println("before:  " + sirius);
     commonName(sirius);
     System.out.println("after: " + sirius);
   }

   public static void commonNamet(Body bodyRef) {
     bodyRef.name = "Dog Star";
     bodyRef = null;
   }
}
```

Using Method to Control Access

```
class Body {
    private long idNum;
    public String name = "<unnamed>";
    public Body orbits = null;
    private static long nextID = 0;
    Body () {
n     idNum = nextID++;
    }
    public long getID() {
     return idNum;
    }
  // ....
}
```

Cont

```
class Body {
    private long idNum;
    public String name = "<unnamed>";
    public Body orbits = null;
    private static long nextID = 0;
   // Constructors
    public long getID() {
     return idNum;
    }
```

Cont

```
        public String getName()
        {return name;}
        public void setName(String newName){
         name = newName;
        }
        public Body getOrbits()
        {return orbits;}
        public boid setOrbits(Body orbitsAround) {
         orbits = orbitsAround;
         }
    }
```

this

```
public Body(String name, Body orbits) {
    this();
    this.name = name;
    this.orbits = orbits;
}
```

Console Input

- reads from `System.in` object which only reads bytes.
- An `InputStreamReader` reads characters

```
InputStreamReader reader =
   newInputStreamReader(System.in);
```

- A `BufferedReader` can read strings.

```
BufferedReader console =
 new BufferedReader(reader);
```

Example

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

/**
   This program tests input
         from a console window.
*/
public class ConsoleInputTest
{
```

Example (cont)

```
public static void main(String[] args)
                        throws IOException
{
Purse myPurse = new Purse();
BufferedReader console = new BufferedReader(
   new InputStreamReader(System.in));

System.out.println
     ("How many nickels do you have?");
String input = console.readLine();
int count = Integer.parseInt(input);
myPurse.addNickels(count);
```

Example (cont)

```
System.out.println
        ("How many dimes do you have?");
input = console.readLine();
count = Integer.parseInt(input);
myPurse.addDimes(count);
System.out.println
         ("How many quarters do you have?");
input = console.readLine();
count = Integer.parseInt(input);
myPurse.addQuarters(count);
double totalValue = myPurse.getTotal();
System.out.println
     ("The total is " + totalValue);
System.exit(0);
}
```