Contents

Indentation, White space .............................................................. 2
Names .......................................................................................... 3
Comments .................................................................................... 6
  Commenting Efficiently............................................................... 7
  Kinds of Comments .................................................................. 10
Keep Code simple ........................................................................ 12
Asserts .......................................................................................... 13
Testing ........................................................................................ 13
Try-catch ..................................................................................... 14
Points & Rectangles ..................................................................... 15

References

Student papers

Code Complete, Steve McConnell, Microsoft Press, 1993,
Assignment 2 Comments
Indentation, White space

Use white space to make program readable

Use indentation to show the program structure

One space indentation is not enough

```java
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
```

```java
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
```

```java
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
```

```java
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
```

```java
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
```
Names

Use full names
Use meaningful name
Follow Java's standards
Be consistent

public class Rectangle {
    Point p1;
    Point p2;
    Point p3;
    Point p4;
    Blah;
}

public class Rectangle {
    Point origin;  //Which origin?
    float width;
    float height;
    Blah
}

public class Rectangle {
    Point lowerLeftCorner;
    float width;
    float height;
    Blah
}
Make sure name matches the usage

Some assignments used the name upperLeft but meant lowerLeft

Java Standards

thisIsTheJavaWayForMethodsAndVariables

ClassName

this_is_not_the_java_way

use equals() not identical() for a test of equality

```java
class Rectangle {
    public boolean containsRectangle( Rectangle ) { blah }
    public boolean containsPoint( Point ) { blah }
}
```

```java
class Rectangle {
    public boolean contains( Rectangle ) { blah }
    public boolean contains( Point ) { blah }
}
```
Set & Get

```java
public class Rectangle {
    public void setOrigin( Point origin ) {blah}
    public Point getPointOrigin( ) {blah}
}

public class Rectangle {
    public void origin( Point origin ) {blah}
    public Point origin( ) {blah}
}
```

**The Java Standard**

```java
public class Rectangle {
    public void setOrigin( Point origin ) {blah}
    public Point getOrigin( ) {blah}
}
```
Comments

"Comments are easier to write poorly than well, and comments can be more damaging than helpful"

Comments should provide information that is not clear from the code

First see if you can make the code clearer

// A Constructor
public Rectangle()
{
    {blah
}

// Method: getCenter
// Operation: computes the center of the rectangle by adding the
//            lowerleft corner and the upper right corner and dividing by 2
// Returns: point that is the center of the rectangle
public Point getCenter()
{
    return lowerLeft.add( upperRight).divide( 2);
}
Commenting Efficiently

• Use styles that are easy to maintain

/*******************************************************************************/
* module: Print
*
* author: Roger Whitney
* date: Sept. 10, 1995
*
* blah blah blah
*
*******************************************************************************/

/*******************************************************************************/
module: Print

author: Roger Whitney
date: Sept. 10, 1995

blah blah blah

*******************************************************************************/

• Comment as you go along
What does this do?

for i := 1 to Num do  
  MeetsCriteria[ i ] := True;
for i := 1 to Num / 2 do begin
  j := i + i;
  while ( j <= Num ) do begin
    MeetsCriteria[ j ] := False;
    j := j + i;
  end;
for i := 1 to Num do
  if MeetsCriteria[ i ] then
    writeln( i, ' meets criteria' );
How many comments does this need?

for PrimeCandidate:= 1 to Num do
    IsPrime[ PrimeCandidate] := True;

for Factor:= 1 to Num / 2 do begin
    FactorableNumber := Factor + Factor ;
    while ( FactorableNumber <= Num ) do begin
        IsPrime[ FactorableNumber ] := False;
        FactorableNumber := FactorableNumber + Factor ;
    end;
end;

for PrimeCandidate:= 1 to Num do
    if IsPrime[ PrimeCandidate] then
        writeln( PrimeCandidate, ' is Prime ' );

Don't patch poorly written code with comments

Make the code better
Kinds of Comments

• Repeat of the code

\[ X := X + 1 \quad /* \text{add one to } X */ \]

/* if allocation flag is zero */

if ( AllocFlag == 0 ) ...

• Explanation of how code works

Used to explain complicated or tricky code

\[ *(p++)->*c = a \quad /* \text{first we need to increase } p \text{ by one, then ..} */ \]

Make code simpler before commenting

\[ (*(p++))->*c = a \]

ObjectPointerPointer++;  
ObjectPointer = *ObjectPointerPointer;  
ObjectPointer - >*DataMemberPointer = a;
• Marker in the code

    /* **** Need to add error checking here **** */

• Summary of the code

    Distills a few lines of code into one or two sentences

• Description of the code's intent

    Explains the purpose of a section of code

    /*get current employee information */   intent

    /* update EmpRec structure */       what
Keep Code simple
Version 1

if ( a == true && b == true )
    return true
else
    return false

Version 2

return ( a == true && b == true )

Version 3

return a && b

Version 4

return isGreen && isNegative
Asserts

assert( a == true )

verses

assert( a )

assert( a == false )

verses

assert( !a )

Testing

public class Rectangle {
    public boolean contains( Point origin ) {
        return true;
    }
}

The above passes most of your tests for contains

Should use a point:
    Inside
    Outside
    On border
Try-catch

try
{
    blah
}
catch (Exception error)
{
    System.out.println(" There was an error ");
    System.exit( 0 );
}

The above is almost always the wrong thing to do

The above should not be used in libraries
Points & Rectangles

Using points in rectangle class simplifies the implementation

There was no need to add getX getY to the point