CHAPTER4

Methods use instance variables(EXERCISE)

(GETTERS,SETTERS – ACTUATORS,MUTUATORS)

1.SHARPEN YOUR PENCIL

a)int a = calcArea(7, 12);
short c = 7;
calcArea(c, 15);

b)calcArea(2, 3);

2. BE the Compiler

(A)42 84
(B) void getTime() needs to be String getTime() because the string "12345" is returned as time

3.WHO AM I

A class can have any number of these. - method
A method can have only one of these. - return
This can be implicitly promoted. – return
I prefer my instance variables private. - encapsulation
It really means "make a copy." – pass by value
Only setters should update these.- variable
A method can have many of these.- arguments
I return something by definition. - getter
I shouldn't be used with instance variables. - public
I can have many arguments. - method
By definition, I take one argument. - setter
These help create encapsulation.- private
I always fly solo.- return

4.MIXED MESSAGES

(i) i < 9 index < 5 - > **14 1**
(ii) i < 20 index < 5 ->**25 1**
(iii) i < 7 index < 7 -> **14 1**
(iv) i < 19 index < 1 ->**20 1**

5. POOL PUZZLE

```
public class Puzzle4 {
   public static void main(String [] args) {
      Va lue [ ] values = new Value[6];
      Puzzle4 [ ] values = new Puzzle4[6];
      int number = 1;
      int i = 0;
      while (i < 6) {
         values [i] = new Value(i);
         values[i].intValue = i;
         number = number * 10;
         i =i + 1;
      }
      int result = 0;
      i = 6;
      while (i > 0) {
         i=i-1;
         result = result + values[i].doStuff;
      }
      System.out.println("result " + result);
   }
}

class Value(){
   int intValue;
   public int doStuff(int factor) {
      if (intValue > 100) {
         return intValue * factor;
      } else {
         return intValue * (5-factor);
      }
   }
}
```

6. **Five-Minute Mystery** – keeping things private