1 Part One

Sharpen Solutions

Sometimes there's more than one right answer. And sometimes the answer is whatever you want it to be. If you don't like our answers, argue with us. If we're just plain wrong, we'll change it and give you credit on the web site. If *you're* wrong, we'll publicly humiliate you, using a very large font.

Just kidding.

Please share your ideas and solutions with us, and we'll add them with your name (unless you want to be anonymous, and who could blame you.)

Page 4

```
int size = 27;
```
declare an integer variable named 'size' and give it the value 27

```
String name = "Fido";
```
declare a String variable named 'name' and give it the String value "Fido"

```
Dog myDog =
    new Dog(name, size);
```
declare a Dog variable named 'myDog' and give it a new Dog (that has a name and a size)

```
x = size - 5;
```
subtract 5 from the current value of the variable 'size', assign the result to the variable 'x'

```
if (x < 15) myDog.bark(8);
```
if the value of x is less than 15, then tell myDog to bark 8 times

```
while (x > 3) {
   myDog.play();
```
as long as the value of x is greater than 3, tell myDog to play

Page 11

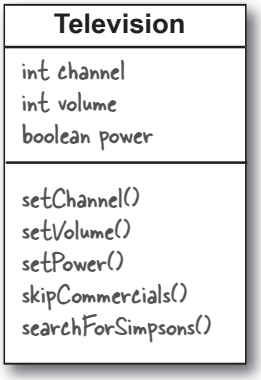
Given the output:

```
% java Test
DooBeeDooBeeDo
```

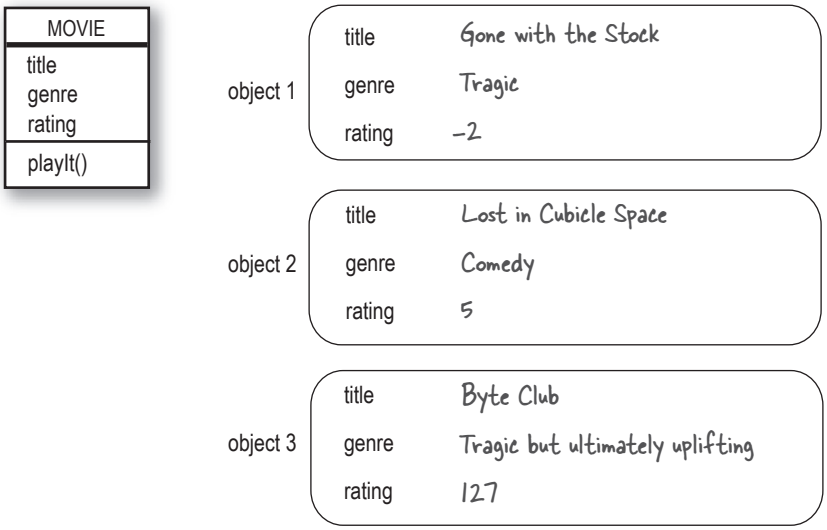
Fill in the missing code:

```
public class DooBee {
  public static void main (String[] args) {
    int x = 1;
    while (x < __3__ ) {
      System.out.__print____("Doo");
      System.out.__print____("Bee");
      x = x + 1;
    }
    if (x == ___3___  ) {
      System.out.print("Do");
    }
  }
}
```

Page 32



Page 35



Page 50

Circle the legal statements from the following list:

```
1. int x = 34.5;

2. boolean boo = x;

3. int g = 17;

4. int y = g;

5. y = y + 10;

6. short s;

7. s = y;

8. byte b = 3;

9. byte v = b;

10. short n = 12;

11. v = n;

12. byte k = 128;

13. int p = 3 * g + y;
```

Page 50

What is the current value of pets[2]? null

What code would make pets[3] refer to one of the two existing Dog objects?

pets[3] = pets[0]

Page 85

What's legal?

Given the method below, which of the method calls listed on the right are legal?

Put a checkmark next to the ones that are legal. (Some statements are there to assign values used in the method calls).

```
int calcArea(int height, int width) {
   return height * width;
}
```

```
int a = calcArea(7, 12);
short c = 7;
calcArea(c,15);
int d = calcArea(57);
calcArea(2,3);
long t = 42;
int f = calcArea(t,17);
int g = calcArea();
calcArea();
byte h = calcArea(4,20);
int j = calcArea(2,3,5);
```

need two args

't' is a long (too big for the int parameter 'height')

need two args

need two args

calcArea returns an int, not a byte

need two args

Page 101

In the next couple of pages we implement the SimpleDotCom class, and then later we return to the test class. Looking at our test code above, what else should be added? What are we *not* testing in this code, that we *should* be testing for? Write your ideas (or lines of code) below:

Make a fake user guess that is a MISS instead of a hit

Try all three hits

Try out a range of guesses

Try duplicate guesses

(these are just a few...)

Page 105

Turn the to the next page in your book (106) for the answer. But then, you obviously know that already. We just put this in for completeness. Didn't want you thinking we just skipped it. Although we actually *are* skipping it. Here in the solutions document, anyway, not in the real book. You know what we mean.

Page 111

It's a cliff-hanger!

Will we ***find*** the bug?

Will we ***fix*** the bug?

Will Ben marry J-Lo?

Stay tuned for the next chapter, where we answer these questions and more...

And in the meantime, see if you can come up with ideas for what went wrong and how to fix it.

The current version of the game cares only about the NUMBER of hits, not what the actual hits really are. So entering the same number (as long as it's a hit) three times is the same as entering the three correct, but different, numbers corresponding to the three hit locations.

So, we need to somehow keep track of each hit, and maybe "cross it off the list" to show that it's been hit and is no longer a valid entry.

Page 130

Turn to page 132 in your book for the answer.

Page 141

We didn't do an answer for this one, but nobody around here can remember why. Must have been some excuse about how that makes it more of a "learning opportunity" for you. If you've got an answer you want to "share" with others (for that warm fuzzy feeling and good karma points), let us know and we'll include it (with your name).

[Kathy, this looks pretty weak here for chapter six. Doesn't look like we're giving them ANYTHING! Don't you feel guilty about that? —Bert]

[No. —Kathy]

Page 165

How many instance variables does Surgeon have? 1

How many instance variables does FamilyDoctor have? 2

How many methods does Doctor have? 1

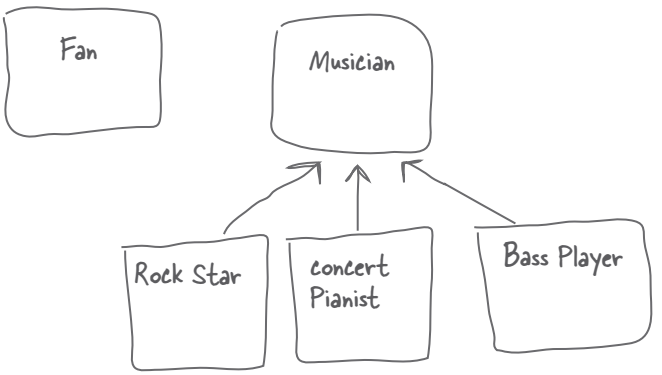
How many methods does Surgeon have? 2

How many methods does FamilyDoctor have? 2

Can a FamilyDoctor do treatPatient()? Yes

Can a FamilyDoctor do makeIncision()? No

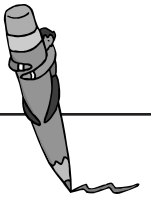
Page 172



I think I see a problem here... what if you have a bass player who IS a rock star? A rock star who IS a bass player? Not all bass players are rock stars, and not all rock stars are bass players, so what do you do?

[answer: you'll have to wait for chapter 8 where we talk about interfaces...]

Page 175



Put a check next to the relationships that make sense.

- ☐ **Oven extends Kitchen**
- ☑ **Guitar extends Instrument**
- ☐ **Person extends Employee**
- ☐ **Ferrari extends Engine**
- ☑ **FriedEgg extends Food**
- ☑ **Beagle extends Pet**
- ☐ **Container extends Jar**
- ☐ **Metal extends Titanium**
- ☑ **GratefulDead extends Band**
- ☑ **Blonde extends Smart**
- ☐ **Beverage extends Martini**

What if I want Beagle to extend Dog, but not all dogs are pets? (chapter 8)

Hint: apply the IS-A test

Page 201

Concrete	Sample class	Abstract
golf course simulation	Tree	tree nursery application
monopoly game	House	architect application
satellite photo application	Town	Atlas/map application
video game	Football Player	coaching application
banquet seating app	Chair	home floorplan design app
business modeling app	Customer	technical support app
employee training app	Sales Order	Store inventory system
home inventory program	Book	online book store
Mall store directory kiosk	Store	Warehouse distribution system
Simple business accounting	Supplier	Just-in-time inventory system
simple golf game	Golf Club	Pro shop POS system
Parts Inventory app	Carburetor	Engine design software
Home / architectural design	Oven	Gourmet restaurant app

Note: this is a little confusing and a little subjective, but here's a tip for this exercise -- the abstract category is for applications where the class in the center column would need to be SUBCLASSED. The concrete category is for applications where the class in the center can be concrete, and the only thing differentiating one instance from another is instance variable values. For example, in a home inventory system, you probably don't need to distinguish between different classes of books, you simply need an instance variable with the title, so that each instance of Book represents a book in your house. But a bookstore might *care* about different TYPES of books like FictionBook, TradePaperback, Children, etc. since each type might have different policies for stocking, discounts, and advertising.