Types of Questions

Here are descriptions, with some examples, of the types of questions that may appear on the common final.

> These examples are intended to augment the verbal descriptions of the types of questions. The actual questions that end up on the test will of course be different, but will be intended to cover the same material.

Sample answers have been given for each question.

Question Category 1: given some Java code, execute it

For any code fragment, method body, or full program using any part of our official Java Subset, students should be able to trace its execution.

> Each instructor should present a model of Java semantics that allows students to execute code by hand, drawing "memory pictures" that capture storage of values in various kinds of memory cells, including static variables, instance variables within objects, and passing of arguments in for parameters.
>
> The semantic model presented should include treatment of references enabling tracking variables whose reference contents change, multiple references to the same object, behavior of `==`, `equals`, and `compareTo`, and automatic invocation of `toString`.

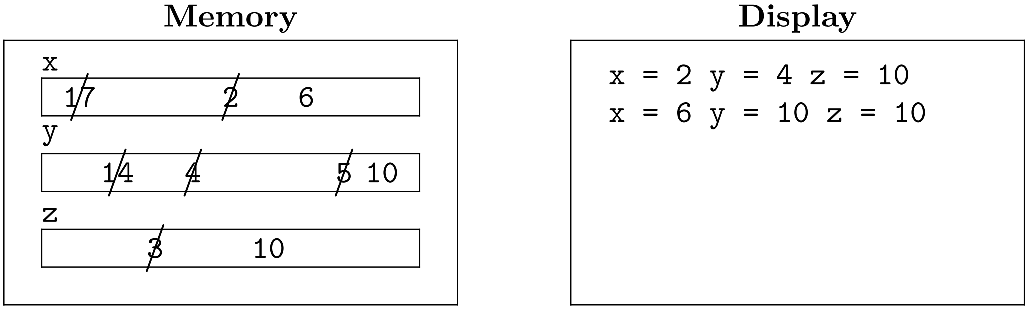
1.1 Students should be able to evaluate expressions of reasonable complexity, given values of variables involved, showing mastery of issues such as operator precedence, `int` arithmetic, and assignment operations.

Example Question:

Trace execution of this code fragment, showing contents of memory and output in the space provided:

```
int x = 17;
int y = 14;
int z = 3;
y /=  z;
x %= 3;
z = x + 2 * y;
System.out.println("x = " + x + " y = " + y + " z = " + z);
x += y;
y++;
y *= 2;
System.out.println("x = " + x + " y = " + y + " z = " + z);
```

Example Solution:



```
x = 2 y = 4 z = 10
x = 6 y = 10 z = 10
```

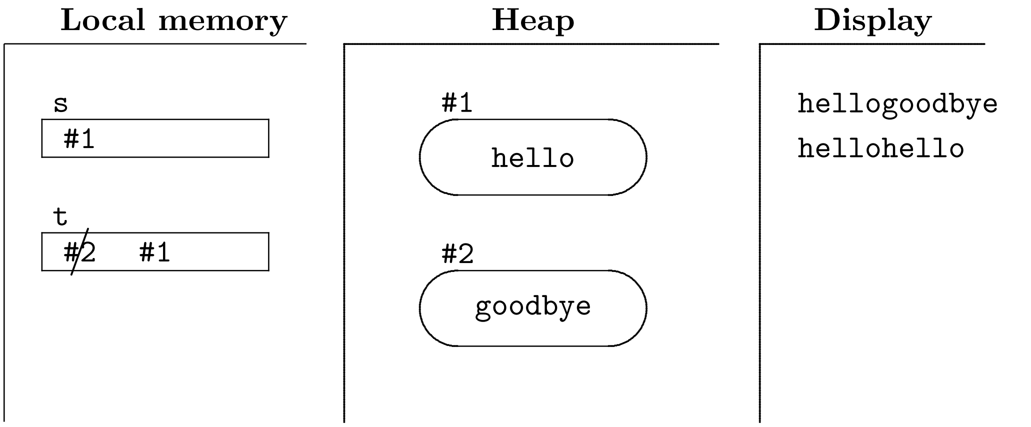
1.2 Students should be able to trace execution of code that assigns references to objects.

Example Question:

Trace execution of this code fragment, showing contents of memory and output in the space provided:

```
String s = "hello";
String t = "goodbye";
System.out.println(s + t);
t = s;
System.out.println(s + t);
```

Example Solution:



```
hellogoodbye
hellohello
```

1.3 Students should be able to trace method calls, using given arguments and producing the output value.

Example Question:

Trace execution of this class, beginning with `main`, showing contents of memory for each method call, and showing the output to the display:

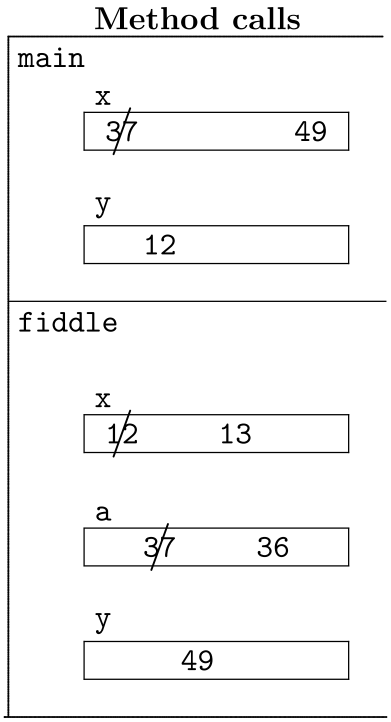
```
public class Cat1c {
    public static void main(String[] args) {
        int x = 37;
        int y = 12;

        x = fiddle(y, x);
        System.out.println("x = " + x + " y = " + y);
    }

    public static int fiddle(int x, int a) {
        int y;

        y = x + a;
        x++;
        a--;
        System.out.println("x = " + x + " y = " + y + " a = " + a);
        return y;
    }
}
```

Example Solution:



Display

```
x = 13 y = 49 a = 36
x = 49 y = 12
```

1.4 Students should be able to trace execution of relatively simple code involving nested branching and looping statements, with use of logical and relational operators.

Example Question:

Trace execution of this code fragment, showing contents of memory and output in the space provided:

```
int x=10, y=11, z=3;
do {
    System.out.println(x + " " + y + " " + z);
    if (y > z) {
        y = y - 5;
    }
    else {
        y = y + 3;
    }
    x = x - 2;
} while (x > 0);
System.out.println("Bye");
```

Example Solution:

Local memory

x

~~10~~ ~~8~~ ~~6~~ ~~4~~ ~~2~~ 0

y

~~11~~ ~~6~~ ~~1~~ ~~4~~ ~~-1~~ 6

z

3

Display

```
10 11 3
8 6 3
6 1 3
4 4 3
2 -1 3
Bye
```

1.5 Students should be able to trace code, typically involving loops, with both one and two dimensional arrays and `ArrayList` instances.

Example Question:

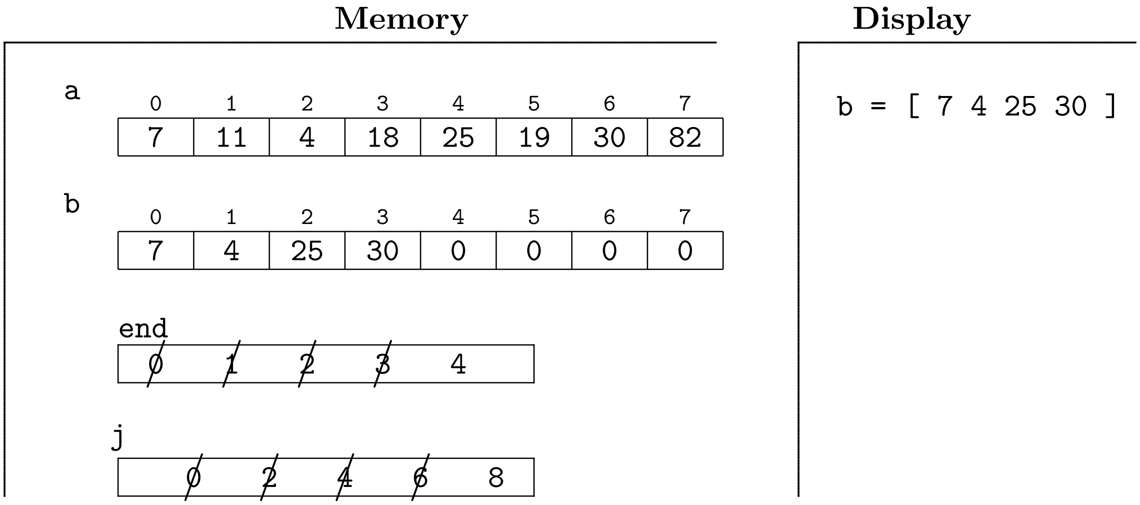
Trace execution of this program, beginning with `main`, showing contents of memory and output in the space provided:

```
public class Cat1e{
    public static void main(String[] args){
        int[] a = { 7, 11, 4, 18, 25, 19, 30, 82 };
        int[] b = new int[ a.length ];

        int end = 0;
        for (int j=0; j<a.length; j+=2) {
          b[ end ] = a[ j ];
          end++;
        }

        System.out.print("b = [");
        for (int k=0; k<end; k++) {
            System.out.print(b[k] + " ");
        }
        System.out.println(" ]");
    }
}
```

Example Solution:



Memory

a

0	1	2	3	4	5	6	7
7	11	4	18	25	19	30	82

b

0	1	2	3	4	5	6	7
7	4	25	30	0	0	0	0

end

~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

j

~~0~~ ~~2~~ ~~4~~ ~~6~~ 8

Display

```
b = [ 7 4 25 30 ]
```

1.6 Students should be able to trace code that constructs several instances of a given class and calls some instance methods.

Example Question:

Trace execution of this program, beginning with `main`, showing contents of memory and output in the space provided:

```
public class Box {
    private int x, y, w, h;

    private static int maxWidth = 6;

    public Box(int xIn, int yIn) {
        if ( xIn > maxWidth ) {
            x = maxWidth;
        }
        else {
            x = xIn;
        }
        y=yIn;
        w=0;
        h=3;
    }

    public void grow(int a) {
        w += a;
        h += a;
    }

    public void fight( Box other ) {
        x  = other.y + 5;
        other.x = 7;
    }

    public int cross(){
        return x * y;
    }

    public String toString(){
        return "[(" + x + "," + y + ") " + w + " " + h + "]";
    }

    public static void main(String[] args) {
        Box a = new Box(12, 5);
        Box b = new Box(2, 7);
        System.out.println("new boxes: " + a + " " + b);
        a.grow(3);
        b.grow(2);
        System.out.println("after grow: " + a + " " + b);
        b.fight(a);
        System.out.println("after fight: " + a + " " + b);
        System.out.println("cross: " + a.cross() );
    }

}
```

Example Solution:

Static Variables

maxWidth

6

Heap

#1

x

~~6~~ 7

y

5

w

~~0~~ 3

h

~~3~~ 6

#2

x

~~2~~ 10

y

7

w

~~0~~ 2

h

~~3~~ 5

Display

```
new boxes:  [(6,5) 0 3] [(2,7) 0 3]

after grow:  [(6,5) 3 6] [(2,7) 2 5]

after fight:  [(7,5) 3 6] [(10,7) 2 5]

cross:  35
```

1.7 Students should be able to trace code that does file input/output, including behavior of `try-catch` statements.

Example Question:

Suppose that a data file named `test3` has these contents:

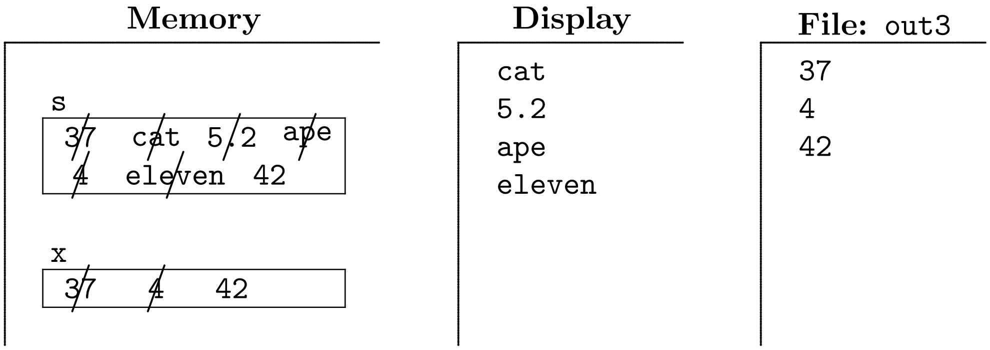
```
37 cat 5.2
ape 4 eleven
42
```

Trace execution of the program listed below, beginning with `main`, showing contents of memory, output to the console, and the contents of the file `out3`:

```java
import java.io.File;
import java.util.Scanner;
import java.io.PrintWriter;

public class Cat1f {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("test3"));
            PrintWriter output = new PrintWriter("out3");
            do {
                String s = "";
                try {
                    s = input.next();
                    int x = Integer.parseInt(s);
                    output.println(x);
                }
                catch(Exception e) {
                    System.out.println(s);
                }
            } while(input.hasNext());
            input.close();
            output.close();
        }
        catch(Exception e2) {
            System.out.println("File not found");
        }
    }
}
```

Example Solution:



Memory

s

~~37~~ ~~cat~~ ~~5.2~~ ~~ape~~ ~~4~~ ~~eleven~~ 42

x

~~37~~ ~~4~~ 42

Display

```
cat
5.2
ape
eleven
```

File: `out3`

```
37
4
42
```

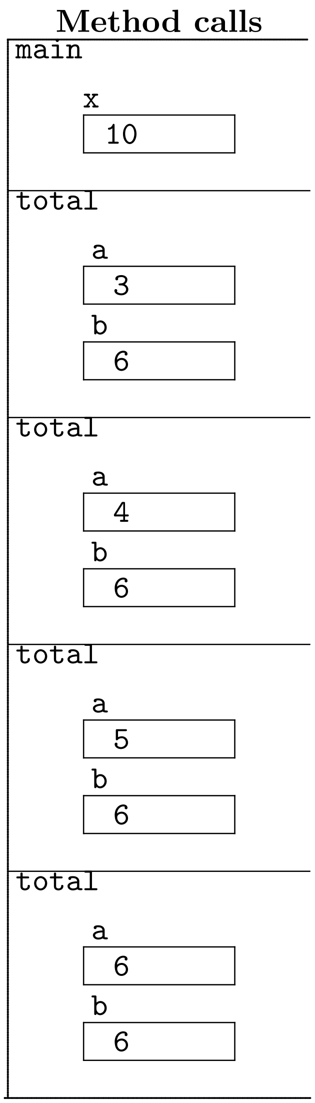
1.8 Students should be able to trace code that performs recursive method calls.

Example Question:

Trace execution of this program, beginning with `main`, showing contents of memory and output in the space provided:

```
public class Cat1g {
    public static void main(String[] args) {
        int x = total(3, 6);
        System.out.println("x = " + x);
    }
    public static int total( int a, int b ) {
        System.out.println(a + " " + b);
        if (a < b) {
            return a + total(a+1, b);
        }
        else {
            return 0;
        }
    }
}
```

Example Solution:



Display

```
3 6
4 6
5 6
6 6
x = 12
```

Question Category 2: given a natural language description of some behavior, write Java code that produces it

> Given behavior that requires either fairly simple algorithmic design for a novel situation, or requires use of one of the algorithms specified below, students should be able to write code that implements that behavior.

2.1 Students should be able to write Java code that closely corresponds to an algorithm given in English or otherwise described in detail, say with a diagram, chart, or storyboard.

Example Question:

Create a complete Java application in a class named `StrangeCalc` that behaves as a strange calculator as follows. This calculator maintains a "current value" which is an integer. The user repeatedly enters commands. Each command performs some operation on the current value, typically changing it, and then the new current value is displayed. The current value starts at 0.

Here are the commands:

Command	Operation
twice	Replace current value by twice the current value
hop	Replace the current value by three times the current value plus one
shrink	Replace the current value by half the current value (using integer division)
quit	Halt the application

Here is a screen image showing a sample run of the program:

```
value: 0
? hop
value: 1
? hop
value: 4
? twice
value: 8
? hop
value: 25
? shrink
value: 12
? quit
```

Example Solution:

```
import java.util.Scanner;

public class StrangeCalc {

    public static void main(String[] args) {

        Scanner keys = new Scanner( System.in );
        int value = 0;
        String command;

        do {

            System.out.println("value: " + value);
            System.out.print("? ");
            command = keys.nextLine();

            if (command.equals("twice")) {
                value *= 2;
            }
            else if (command.equals("hop")) {
                value = 3 * value + 1;
            }
            else if (command.equals("shrink")) {
                value /= 2;
            }

        } while(!command.equals( "quit" ));

    }

}
```

2.2 Students should be able to write method bodies for instance methods, given code implementing the instance variables and documentation for the desired behaviors of the methods.

Example Question:

Suppose you are starting work on a program to be used for a restaurant. As part of this work, you realize you need to have a class `Order` to store and work with the items a table of customers orders.

The restaurant only sells burgers, fries, and drinks. An order has instance variables storing how many burgers, how many sacks of fries, and how many drinks the table has ordered, along with the name of the staff person who took the order.

The restaurant encourages people to order “meals,” where a meal consists of one burger, one sack of fries, and one drink. So, if a table orders 5 meals, their order will have 5 burgers, 5 sacks of fries, and 5 drinks.

But, after ordering their meals, they can also adjust the number of each item they want (only adjusting burgers is asked for—similar methods for fries and drinks would be needed).

Each burger costs $3.50, each sack of fries costs $2.75, and each drink costs $2.25.

Write the bodies for all the constructors and instance methods, following the documentation for each method and using the given instance variables.

```
public class Order {
    private String name;  // the server who took the order
    private int burgers;  // the number of burgers in the order
    private int fries;  // the number of sacks of fries in the order
    private int drinks;  // the number of drinks in the order

    //  given a server’s name and a number of meals,
    //  construct this order
    public Order(String server, int numMeals) {
    }
    // add the given number of burgers to the order
    // (n can be negative to remove burgers)
    // Don’t let the number of burgers go below 0
    public void addBurgers(int n) {
    }
    //  compute and return the cost of this order
    public double figureCost() {
    }
    // return whether this order and the given other order
    // have the same server
    public boolean haveSameServer(Order other) {
    }
}//  Order
```

Example Solution:

```
public class Order {

    private String name;  // the server who took the order
    private int burgers;  // the number of burgers in the order
    private int fries;  // the number of sacks of fries in the order
    private int drinks;  // the number of drinks in the order

    //  given a server's name and a number of meals,
    //  construct this order
    public Order(String server, int numMeals) {
        name = server;
        burgers = numMeals;
        fries = numMeals;
        drinks = numMeals;
    }

    // add the given number of burgers to the order
    // (n can be negative to remove burgers)
    // Don't let the number of burgers go below 0
    public void addBurgers(int n) {
        burgers += n;
        if (burgers < 0) {
            burgers = 0;
        }
    }

    //  compute and return the cost of this order
    public double figureCost() {
        return 3.5 * burgers + 2.75 * fries + 2.25 * drinks;
    }

    // return whether this order and the given other order
    // have the same server
    public boolean haveSameServer(Order other) {
        return name.equals(other.name);
    }

}//  Order
```

2.3 Students should be able to write modified versions of the following core algorithms to implement the desired behavior:

- swap the contents of two memory cells, typically contained in an array or `ArrayList`
- traverse a list and perform some method on each item
- traverse a list and accumulate information
- traverse a list and find an item with some desired property
- traverse a list and find a minimum/maximum item

Example Question:

Suppose that the `Puppy` class has a method with signature:

`public int cuteness()`

that returns the cuteness of a puppy.

Your job on this question is to write the body of the method documented below.

```
//  return the total cuteness of
// all the puppies in the given list
public int getTotalCuteness( ArrayList<Puppy> list ){
}
```

Example Solution:

```
//  return the total cuteness of
// all the puppies in the given list
public int getTotalCuteness( ArrayList<Puppy> list ){
    int total = 0;

    for (int k=0; k<list.size(); k++) {
      total += list.get(k).cuteness();
    }
}
```

2.4 Students should be able to write code that shifts items in an array or `ArrayList` to remove an item or insert a new item

Example Question:

Write the body of the method given below, following the documentation:

```
// Copy every item stored in the given array three positions
// toward the beginning, but don’t copy any items that
// would cause an exception
// For example, if the method is called with
//    array =  [ "a", "b", "c", "d", "e", "f", "g", "h" ]
// then after the method executes
//    array = [ "d", "e", "f", "g", "h", "f", "g", "h" ]
public static void shiftLeft3(String[] arrray) {
}
```

Example Solution:

```
public static void shiftLeft3(String[] arrray) {
    for(int j=3; j<array.length; j++) {
        array[ j-3 ] = array[ j ];
    }
}
```

2.5 Students should be able to write code that traverses a sub-rectangle of a 2D array, processing the items in some way.

Example Question:

Write the body of the method given below, following the documentation:

```
// add up the items in array in the sub-rectangle
// with rows between firstRows and lastRow, inclusive,
// and columns between firstCol and lastCol, inclusive,
// and return that total
// Assume that the four given integers are valid for the array
public static int totalRegion(int[][] array,
                              int firstRow, int lastRow,
                              int firstCol, int lastCol){
}
```

Example Solution:

```
public static int totalRegion(int[][] array,
                             int firstRow, int lastRow,
                             int firstCol, int lastCol) {
    int total = 0;

    for(int row=firstRow; row<=lastRow; row++) {
        for( int col=firstCol; col<=lastCol; col++) {
            total += array[ row ][ col ];
        }
    }

    return total;
}
```

Question Category 3: given a scenario, design the corresponding aspects of a Java application

3.1 Given a scenario, design the classes required to implement it, where the design includes appropriate class names and instance variables with appropriate data types and names.

Example Question:

Suppose you are starting to create an application that will implement software to manage a simple old-time airline. Your job is to write the classes suggested by the following partial description, with each class having appropriate instance variables

An *airline* will consist of a list of all the current and scheduled flights. A *flight* has a flight number (all flights for the airline for its entire life will be numbered as 1, 2, and so on), a list of seats, a departure airport and arrival airport (each represented by a unique three-letter code like ”DEN” for Denver International Airport), a scheduled departure time, and a scheduled arrival time (each represented by an integer between 0 and 2359 inclusive). A *seat* has a row between 1 and 60, a position which is one of `A` through `F`, and a string which is the passenger's name, or is `"-"` if the seat is unpurchased.

Begin all the relevant classes for this program by writing below the first line of each class—thus saying what name you think it should have—followed by lines of code for each class that create its instance variables, as suggested by the scenario. For each instance variable, be careful to write the most appropriate type and an appropriate identifier.

To show how we want the answer to look, one of the classes—`Airline`–has been completed.

```
public class Airline{
    private ArrayList<Flight> flights;
}
```

Example Solution:

```
public class Airline {
    private ArrayList<Flight> flights;
}

public class Flight {
    private int number;
    private ArrayList<Seat> seats;
    private String departureAirport;
    private String arrivalAirport;
    private int departureTime;
    private int arrivalTime;
}

public class Seat {
    private int row;
    private char position;
    private String passenger;
}
```

3.2 Given a description of the desired behavior of a method, perhaps including pre- and post-conditions, design the method signature.

Example Question:

Write the method signature for the instance method with the documentation given, assuming that the class it will be a member of has this instance variable:

```
    private ArrayList<String> list;
```

```
// given two integers specifying a legal range of indices for list,
// and a target string, remove all occurrences of target in
// the given sublist of list
```

Example Solution:

```
public void removeFromRange( int first, int last,
                             String target )
```

Question Category 4: conceptual questions not necessarily involving programming

4.1 Given a problem and some code or other description of an algorithm to solve it, identify what is wrong with the algorithm, and suggest how to fix it.

Example Question:

Here is a method that is intended to remove all occurrences of the given target from the given list:

```
// remove all occurrences of target from list
public static void remove( ArrayList<String> list, String target ){
    for( int k=0; k<list.size(); k++ ){
        if( list.get(k).equals( target ) ){
            list.remove( k );
        }
    }
}
```

Explain how this algorithm can fail (hint: consider duplicate targets next to each other), and correct the code so that the algorithm is correct.

Example Solution:

If two occurrences of `target` are next to each other, the second one won't be removed.

Corrected code:

```
// remove all occurrences of target from list
public static void remove( ArrayList<String> list, String target ){
    for( int k=0; k<list.size(); k++ ){
        if( list.get(k).equals( target ) ){
            list.remove( k );

            k--;   // <------ after removing an item,
                   //         adjust k to not skip next item

        }
    }
}
```

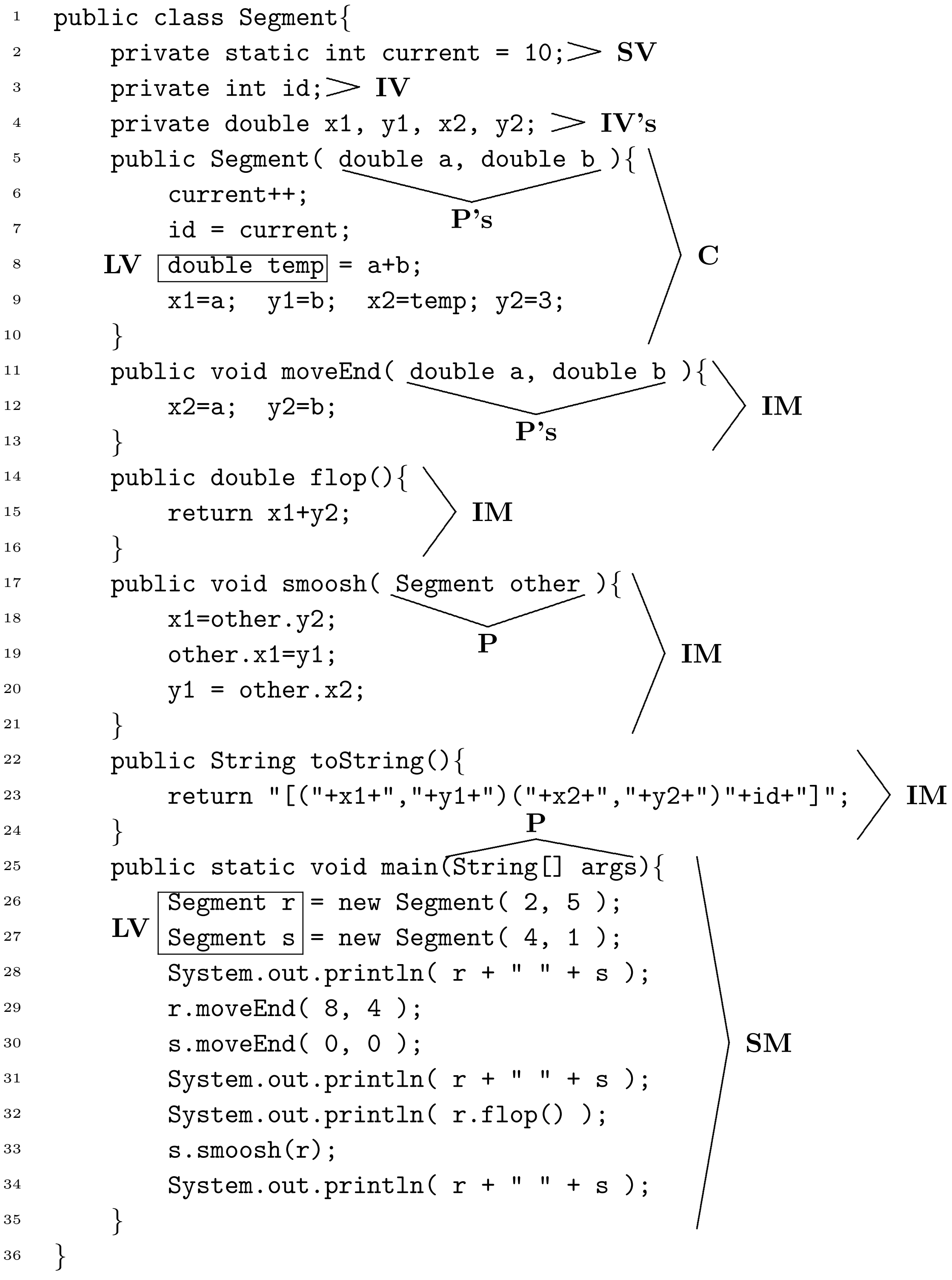
4.2 Given a complete Java class, categorize all its members as *instance variables* (IV), *static variables* (SV), *constructors* (C), *instance methods* (IM), or *static methods* (SM), and categorize all other variable declarations as *parameters* (P) or *local variables* (LV).

Example Question:

On the class listing given below, somehow indicate the code that makes up each class member, and categorize each as `SV`, `SM`, `C`, `IV`, or `IM`. Also, mark all variable declarations that are not class members and label each as `P` or `LV`.

```
public class Segment{
    private static int current = 10;
    private int id;
    private double x1, y1, x2, y2;
    public Segment( double a, double b ){
        current++;
        id = current;
        double temp = a+b;
        x1=a;  y1=b;  x2=temp; y2=3;
    }
    public void moveEnd( double a, double b ){
        x2=a;  y2=b;
    }
    public double flop(){
        return x1+y2;
    }
    public void smoosh( Segment other ){
        x1=other.y2;
        other.x1=y1;
        y1 = other.x2;
    }
    public String toString(){
        return "[("+x1+","+y1+")("+x2+","+y2+")"+id+"]";
    }
    public static void main(String[] args){
        Segment r = new Segment( 2, 5 );
        Segment s = new Segment( 4, 1 );
        System.out.println( r + " " + s );
        r.moveEnd( 8, 4 );
        s.moveEnd( 0, 0 );
        System.out.println( r + " " + s );
        System.out.println( r.flop() );
        s.smoosh(r);
        System.out.println( r + " " + s );
    }
}
```

Example Solution:



```
public class Segment{
    private static int current = 10;
    private int id;
    private double x1, y1, x2, y2;
    public Segment( double a, double b ){
        current++;
        id = current;
        double temp = a+b;
        x1=a;  y1=b;  x2=temp; y2=3;
    }
    public void moveEnd( double a, double b ){
        x2=a;  y2=b;
    }
    public double flop(){
        return x1+y2;
    }
    public void smoosh( Segment other ){
        x1=other.y2;
        other.x1=y1;
        y1 = other.x2;
    }
    public String toString(){
        return "[("+x1+","+y1+")("+x2+","+y2+")"+id+"]";
    }
    public static void main(String[] args){
        Segment r = new Segment( 2, 5 );
        Segment s = new Segment( 4, 1 );
        System.out.println( r + " " + s );
        r.moveEnd( 8, 4 );
        s.moveEnd( 0, 0 );
        System.out.println( r + " " + s );
        System.out.println( r.flop() );
        s.smoosh(r);
        System.out.println( r + " " + s );
    }
}
```

4.3 Given a description of a desired behavior, create a test plan.

Example Question:

Consider the problem of taking three input integers, stored in variables `a`, `b`, and `c`, and rearranging the contents of these variables so that $a \le b \le c$.

Your job on this question is to create a test plan for this problem—write down appropriate test cases, with the desired result for each. For example, one good test case would be written as

```
1 2 3     ---->  1 2 3
```

Your goal is to write down a collection of test cases such that if an algorithm for the problem gave the correct results for each, then you would strongly believe that the algorithm was correct.

Example Solution:

```
1 2 3  ---->  1 2 3
1 3 2  ---->  1 2 3
2 1 3  ---->  1 2 3
2 3 1  ---->  1 2 3
3 1 2  ---->  1 2 3
3 2 1  ---->  1 2 3
1 1 1  ---->  1 1 1
1 1 2  ---->  1 1 2
1 2 1  ---->  1 1 2
2 1 1  ---->  1 1 2
1 2 2  ---->  1 2 2
2 1 2  ---->  1 2 2
2 2 1  ---->  1 2 2
```
