



School of Computer Science & Software Engineering

1st SEMESTER EXAMINATIONS 2003

JAVA PROGRAMMING 124 (230.124)

SURNAME: _____ STUDENT NO: _____

GIVEN NAMES: _____ FACULTY: _____

This paper contains 1 section

This Paper Contains: 27 Pages

Time allowed: 2 Hours

Reading time: 10 Minutes

You may NOT write during reading time

Question 1, which is worth 40 marks, consists of 20 multiple choice parts. Each of these should be answered *in pencil* on the computer-readable multiple choice answer sheet supplied.

Questions 2 – 7, which are worth 10 marks each are code-writing questions and should be answered in the spaces provided on the question paper.

The questions, and parts of questions, are not in any particular order of difficulty, so you should use your reading time to identify those that you find easiest and quickest to answer.

Marks for this paper total 100.

Candidates should attempt all questions.

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.



Instructions

Each question is worth 0 marks.
Marks for this paper total 0.
Candidates should attempt ZERO questions.

1		7	
2		8	
3		9	
4		10	
5		11	
6		12	

1. The first five questions refer to the following source code:

```
/**
 * The Fraction class represents exact rational numbers such as 1/2, 2/5
 * by keeping them in fractional form. It supplies methods for
 * exact arithmetic using fractions.
 * <p>
 * Negative fractions are represented by storing the top as a negative
 * number; the bottom must always be strictly positive.
 *
 * @author Gordon Royle
 * @version 28/05/03
 */

public class Fraction
{
    private int top;
    private int bot;

    public Fraction(int top, int bot) {
        if (bot <= 0)
            throw new IllegalArgumentException("Zero or negative denominator");
        this.top = top;
        this.bot = bot;
        reduce();
    }

    public Fraction add(Fraction other) {
        int newTop = top*other.bot + bot*other.top;
        int newBot = bot*other.bot;
        return new Fraction(newTop, newBot);
    }

    public Fraction multiply(Fraction other) {
        int newTop = top*other.top;
        int newBot = bot*other.bot;
        return new Fraction(newTop, newBot);
    }

    public Fraction reciprocal() {
        if (bot == 0)
            throw new ArithmeticException("Division by zero");
        return (top < 0) ? new Fraction(-bot, -top) : new Fraction(bot, top);
    }
}
```

```
    }

    public void scale(int factor) {
        top = top * factor;
        reduce();
    }

    public double doubleValue() {
        return (double)top / (double)bot;
    }

    private void reduce() {
        if (top == 0)
            bot = 1;
        else {
            int g = (top < 0) ? gcd(-top, bot) : gcd(top, bot);
            top = top / g;
            bot = bot / g;
        }
    }

    private int gcd(int a, int b) {
        if (a < b)
            return gcd(b, a);
        if (b == 0)
            return a;
        return gcd(b, a % b);
    }

    public String toString() {
        return top + "/" + bot;
    }
}
```

- (1) How many `Fraction` objects, in total, are created during the execution of the following sequence of statements?

```
Fraction a;  
Fraction b;  
Fraction c;  
a = new Fraction(11,14);  
b = a.add(new Fraction(2,3));
```

- A. 1
 - B. 2
 - C. 3
 - D. 4
 - E. 5
- (2) What are the values of `a.top` and `a.bot` respectively after the following sequence of statements?

```
Fraction a = new Fraction(4,6);  
Fraction b = a.reciprocal();  
a = a.add(b);
```

- A. 2 and 3
- B. 52 and 24
- C. 13 and 6
- D. 6 and 13
- E. 4 and 6

(3) What is the value of the following expression if `f` currently represents the fraction $5/12$?

```
f.multiply(new Fraction(2,3)).add(f);
```

- A. A (reference to a) new `Fraction` object representing $65/144$.
- B. This expression will not compile.
- C. Evaluating this expression will cause execution to fail with an `IllegalArgumentException`.
- D. A (reference to a) new `Fraction` object representing $5/9$.
- E. A (reference to a) new `Fraction` object representing $25/36$.

(4) What happens when the following sequence of statements is executed?

```
Fraction a = new Fraction(5,8);  
Fraction b = a.reciprocal();  
a.scale(-1);  
System.out.println(b.doubleValue());
```

- A. The string `1.6` appears on the terminal window.
- B. An `IllegalArgumentException` is thrown and execution halts.
- C. The string `-1.6` appears on the terminal window.
- D. The string `-0.625` appears on the terminal window.
- E. The string `-8/5` appears on the terminal window.

(5) What does the following method do?

```
public static Fraction mystery(Fraction[] values) {  
    Fraction total = new Fraction(0,1);  
  
    for (int i=0; i<values.length; i++)  
        total = total.add(values[i]);  
  
    return new Fraction(1,values.length).multiply(total);  
}
```

- A. It returns the sum of all of the `Fractions` in the parameter array.
- B. It returns the average of the `Fractions` in the parameter array.
- C. It returns $1/n$, where n is the number of `Fractions` in the array.
- D. It returns n times the sum of the `Fractions` in the array, where n is the number of `Fractions` in the array.
- E. It returns the product of all the `Fractions` in the parameter array.

(6) Consider the following two expressions, where `x` is an `int`.

```
(3 / x > 10) && (x != 0)
(x != 0) && (3 / x > 10)
```

Which of the following is most correct about evaluating these expressions?

- A. Both expressions can be evaluated for any value of `x`.
 - B. The second expression can always be evaluated, but for some values of `x`, attempting to evaluate the first expression causes an `ArithmeticException` to be thrown.
 - C. The first expression can always be evaluated, but for some values of `x`, attempting to evaluate the second expression causes an `ArithmeticException` to be thrown.
 - D. For any given value of `x`, only one of the two expressions can successfully be evaluated.
 - E. For any given value of `x`, either both expressions can successfully be evaluated, or both will cause an `ArithmeticException` to be thrown.
- (7) If we are searching for an item in a *sorted* array, containing 1024 items, then approximately how many probes must we make in the worst case if we use binary search. (As described in lectures, a “probe” refers to examining an item of the array to see if it equal to the one we are looking for.)
- A. About 11
 - B. About 3
 - C. About 1025
 - D. About 17
 - E. About 22

(8) Suppose an array contains the following numbers (in this order)

10, 2, 4, 6, 3, 11, 1

If you use `INSERTION SORT` to sort the array into increasing order, then after the first *three stages* of the algorithm, what does the array contain?

- A. 2, 10, 4, 6, 3, 11, 1
- B. 2, 4, 5, 3, 1, 10, 11
- C. 2, 4, 10, 6, 3, 11, 1
- D. 1, 2, 10, 4, 6, 3, 11
- E. 1, 2, 3, 10, 4, 6, 11

- (9) The Java class library documentation gives the following description of a method in the class `Integer`

toString

```
public static String toString(int i)
```

Returns a new `String` object representing the specified integer. The argument is converted to signed decimal representation and returned as a string.

Parameters:

`i` - an integer to be converted.

Returns:

a string representation of the argument in base 10.

Suppose that `n` is an `int` whose value you want to draw onto a `Canvas`; for this you need a `String` representing the value of `n`. Which of the following expressions gives you the appropriate `String`?

- A. `n.toString(Integer)`
 - B. `n.toString(n)`
 - C. `n.toString()`
 - D. `Integer.toString(n)`
 - E. `(String) n`
- (10) What is the effect of the following Java statement?

```
Die[] dice = new Die[6];
```

- A. A `Die` object representing a 6-sided die is formed.
- B. An array containing 6 newly constructed `Die` objects is formed.
- C. A reference for an array is created, and initialized to the default value `null`.
- D. Six references, each referring to an array of objects of type `Die` are created.
- E. An array containing 6 object references, each equal to `null`, is formed.

- (11) The class `java.awt.Color` has a class variable called `red` (to represent a red colour). How many variables called `red` are created in a program that uses colours (assuming that no other class in the program has a variable called `red`)?
- A. None.
 - B. One.
 - C. The total number of `java.awt.Color` objects created.
 - D. Once for each time it is referred to in a method that is actually executed.
 - E. Once for each method that refers to it (even if that method is not actually executed while the program is run).

- (12) The Java library class `java.awt.geom.Point2D.Double` has a constructor that takes two `double` values as its arguments, representing the x and y co-ordinates of a point.

Suppose the following piece of code is executed:

```
java.awt.geom.Point2D.Double p1;  
java.awt.geom.Point2D.Double p2;  
p1 = new java.awt.geom.Point2D.Double(1.5, 2.5);  
p2 = new java.awt.geom.Point2D.Double(1.5, 2.5);
```

What is the value of the expression `p1 == p2`, and why?

- A. `true` because `p1` and `p2` represent the same point.
 - B. `false` because `p1` and `p2` are different objects, even though they represent the same point.
 - C. `false` because round-off error occurs when attempting to store 1.5 and 2.5 and so the points are actually slightly different.
 - D. This causes a compiler error because reference types must be compared using `equals` and it is a syntax error to try to use `==`.
 - E. `true` because the compiler recognises that they are the same, so only creates a single object referenced by both `p1` and `p2`.
- (13) Which one of the following statements is true?
- A. The source code for a class can explicitly define zero, one or more constructors.
 - B. The source code for a class must explicitly define exactly one constructor.
 - C. The source code for a class must explicitly define at least one constructor.
 - D. The source code for a class can explicitly define at most one constructor.
 - E. The source code for a class should never contain any explicitly defined constructors.

(14) What is the value of `counter` after the following statements?

```
int counter = 0;

for (int x = 3; x <= 7; x = x + 4) {
    for (int y = 1; y < x; y++) {
        counter = counter + (x*y);
    }
}
```

- A. 9
- B. 50
- C. 156
- D. 214
- E. 868

(15) Consider the following method `int mystery(int a, int b)`, where you may assume that $b \geq 0$.

```
public int mystery(int a, int b) {

    if (b == 0) {
        return 0;
    } else {
        return a + mystery(a, b-1);
    }

}
```

What does this method calculate?

- A. It is a recursive way to calculate $b!$.
- B. It is a recursive way to calculate $a \times b$.
- C. It is a recursive way to calculate b^a .
- D. It is a recursive way to calculate a^b .
- E. It is a recursive way to calculate $a + b$.

(16) Suppose that we are attempting to sort the following array into *increasing* order:

{10, 15, 2, 12, 3, 11, 7}

If we use QUICKSORT, then what does the array contain after the first “partitioning” stage (assume that the fence is chosen to be the first element of the array).

- A. {2, 3, 7, 10, 15, 12, 11}
- B. {10, 2, 3, 7, 15, 12, 11}
- C. {15, 12, 11, 10, 2, 3, 7}
- D. {3, 7, 2, 10, 12, 11, 15}
- E. {2, 3, 7, 10, 11, 15, 12}

(17) Which of the following statements about methods are true? (In this context, a constructor is not considered a method)

1. A class can have two or more methods with the same name.
2. The arguments of a method in a class X can be of any type except X itself.
3. An instance method can call any other methods of the same object, including itself.

- A. Only 1 and 3 are true
- B. Only 2 and 3 are true
- C. Only 1 and 2 are true
- D. Only 1 is true
- E. Only 3 is true

(18) What are the values of **a** and **b** after the following loop?

```
int a = 0;
int b = 0;

for (a=0; a<10 && b<20; a=a+2) {
    a = a + b;
    b = b + a + 1;
}
```

- A. a is 10, and b is 16
- B. a is 28, and b is 45
- C. a is 28, and b is 16
- D. a is 30, and b is 45
- E. a is 12 and b is 16

(19) If somebody's height in inches is stored in a variable `h`, then which of the following commands will correctly translate this into US-style feet-and-inches terminology (one foot contains 12 inches, so somebody who is 74 inches tall is usually said to be "6 feet and 2 inches").

- A. `int feet = h/12;`
`int inches = (h - feet*12)/12;`
- B. `int feet = h%12;`
`int inches = h;`
- C. `double feet = h/12;`
`int inches = h%12;`
- D. `int feet = h/12;`
`int inches = h%12;`
- E. `int feet = h-12;`
`int inches = h/12;`

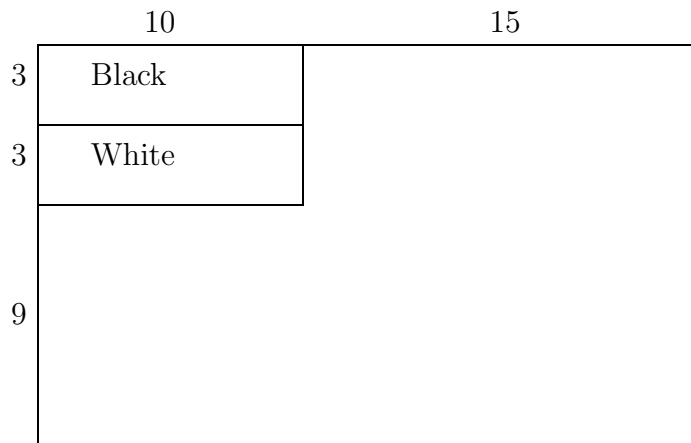
(20) What is the value of `ds(9897)` if `ds` is defined as follows:

```
public static int ds(int n) {  
  
    if (n < 10)  
        return n;  
  
    int m = 0;  
    while (n > 0) {  
        m = m + n % 10;  
        n = n / 10;  
    }  
  
    return ds(m);  
  
}
```

- A. 6
 - B. 7989
 - C. 33
 - D. 10
 - E. 3
-

2.

The flag of *Georgia* has a dark-red background with a small black-and-white stripe in the top-left corner as follows:



The numbers indicate the relative proportions of each segment of the flag.

Write a method `public void drawGA()` that will cause the flag to be drawn on the screen. Use the `Canvas` class from this year's lectures and laboratories, and make sure that the displayed flag is 150 pixels high. Your method must correctly construct any canvases and colours that you need to use. (Hint: A dark-red colour has only 80% of the red component of pure red.)

PLEASE PUT YOUR ANSWER ON THE NEXT PAGE

ANSWER THE GEORGIAN FLAG QUESTION HERE

3.

The *Lucas numbers* are defined similarly to the Fibonacci numbers, but with a different base case:

$$\begin{aligned}L_1 &= 2, \\L_2 &= 1, \\L_n &= L_{n-1} + L_{n-2}.\end{aligned}$$

Therefore the sequence of numbers starts 2, 1, 3, 4, 7, 11, ...

- (a) Write a *recursive* method `public long lucas(int n)` that will return the n 'th Lucas number.

(b) Consider the class `Person` defined (partially) as follows:

```
public class Person {  
  
    private String name;  
    private Person[] children;  
  
    // Constructors and Methods Omitted  
  
}
```

The variable `children` is an array of all the children of this `Person`, and is `null` if this `Person` has no children.

Write a method `public int numDescendants()` that returns the *total* number of descendants of a `Person`. (That is, the number of children plus grand-children plus great-grand-children and so on.)

4.

- (a) Write a method `public int max(int[] a)` that returns the maximum integer in the parameter array `a`.

(b) The class `String` has a method

```
public int length()
```

that returns the *length* of a `String` (i.e. the number of characters in the `String`).

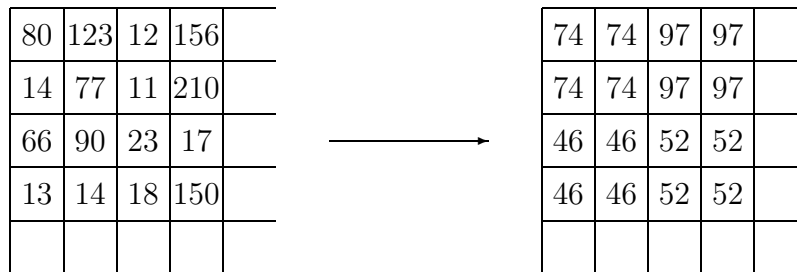
Write a method `public String longestString(String[] names)` that returns the longest `String` in the parameter array `names`. (If there is more than one longest name, return the one that occurs earlier in the array.)

5.

Consider a class `Picture` that represents the data for a grey-scale image. Assume that this class has an array `int[][] pixels` as a private instance variable, storing the pixel values, in such a way that `pixels[i][j]` is the grey-scale value of the (i, j) -pixel (in Java graphics co-ordinates).

The operation of *pixellation* produces a “blocky” effect on a picture by dividing the picture into $k \times k$ blocks of pixels, and replacing all the pixels in each block with the average value of the pixels in that block.

Here is an example where each block has size 2×2 pixels:



Write a method

```
public void pixellate(int k)
```

that will pixellate the `Picture` using blocks of size k .

Your method need only work if k is a positive integer, and the number of rows and columns of the `Picture` are exactly divisible by k . You should check that this precondition is met, and take appropriate action otherwise.

Remember that your method should *only* affect the underlying `Picture` data — no attempt should be made to *display* the `Picture`.

PLEASE PUT YOUR ANSWER ON THE NEXT PAGE

ANSWER THE PICTURE QUESTION HERE

6.

The value π can be approximated by an infinite series as follows:

$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots$$

- (a) Write a method `public double pi1(int n)` that approximates π using the first n terms of this series. (Hint: The class `Math` has a static method `sqrt`.)

(5)

- (b) Write a method `public double pi2(double smallestTerm)` that approximates π using the terms in the series that are *greater than or equal to* `smallestTerm`. (In other words, you stop adding up the terms as soon as they become smaller than the parameter `smallestTerm`.)

(5)

7.

Your company is currently working on a project for a personal financial management system. You have been asked to design and implement a class **Mortgage** to represent a home loan.

Your class must store the current loan amount and have methods that allow the application to:

- Find out the current outstanding balance
- Make a payment, reducing the outstanding balance
- Add one month's worth of interest, increasing the outstanding balance

On the next page, give the Java source code for the *entire class*. Marks will be distributed as follows:

- | | |
|------------------------------------|-----|
| (a) The instance/class variable(s) | (2) |
| (b) The constructor(s) | (2) |
| (c) The outstanding balance method | (2) |
| (d) The make-a-payment method | (2) |
| (e) The add-interest method | (2) |

ANSWER THE MORTGAGE QUESTION HERE

BLANK PAGE FOR ROUGH WORK

BLANK PAGE FOR ROUGH WORK

BLANK PAGE FOR ROUGH WORK

BLANK PAGE FOR ROUGH WORK
