COMP-202 – Introduction to Computing 1
Midterm Exam

March 3rd, 2005

Examiners: Marc Lanctot, Ernesto Posse, Alex Batko

Associate examiner: Joseph Vybihal

Last name _____

First name _____

Id number _____

Section ☐ 1 (Marc Lanctot) ☐ 2 (Ernesto Posse) ☐ 3 (Alex Batko)

Instructions

- No notes, notebooks, textbooks, calculators, cell phones, pagers, laptops or handheld computers are allowed in this exam.
- All answers should be written on the exam sheets. If you need additional space to answer sections 2 and 3, you may write your answers on the back of the exam pages, but you must clearly indicate where your answers are located. **For section 1 (multiple choice) mark your answer in the front page. For section 2 (programming) indicate whether you are answering question 1 or 2.**
- **Read the questions carefully.**
- Language translation dictionaries are permitted.
- Answers may be given in either English or French.

Grading

Section 1: Multiple choice — Section 1 mark: / 20

Question	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Your answer							
Mark	/ 2	/ 3	/ 3	/ 3	/ 3	/ 3	/ 3

Section 2: Short problems — Section 2 mark: / 30

Question	Q1	Q2	Q3
Mark	/10	/10	/10

Section 3: Programming — Section 3 mark: / 50

Question	Q1 or Q2	Q3
Mark	/25	/25

Total: / 100

1 Multiple choice

Question 1 Which of the following statements is illegal in Java?

a) `int i = 1729;`
b) `String name = "Paul";`
c) `boolean warm = winter && temperature < -20.0;`
d) `boolean fun = 1;`
e) `double volume = -19.0;`

Answer d)

Question 2 What will the following program fragment print?

```
int x = 3;
double y = 2.0;
System.out.println( (x-1) / x * 4 + 2 / y );
```

a) 3.666666666666
b) 1.0
c) 3.333333333333
d) 3.0
e) `(x-1) / x * 4 + 2 / y`

Answer b)

Question 3 What will be the values of a, b and c after executing this fragment?

```
double a = 0.0, b = 1.0, c = 0.5;
if (a < b)
{
    if (a < c)
    {
        c = a;
    }
}
else
{
    if (b < c)
    {
        c = b;
    }
}
```

a) 0.0, 1.0, 0.5
b) 0.5, 0.5, 0.5
c) 0.0, 1.0, 0.0
d) 0.5, 1.0, 0.5
e) Nothing, there is an error

Answer c)

Question 4 Which of the following is false?

a) In Java there are two kinds of data: primitive data and objects.
b) References to objects can be used to invoke methods.
c) The attributes of a class must all be of a primitive data-type.
d) An object is a particular instance of a class.
e) Constructors must be named the same as their class's name and don't have return types.

Answer c)

Question 5 Which of the following code fragments will swap the values of two integer variables `a` and `b`?

a)
```
a = b;
b = a;
```

b)
```
a = temp;
temp = b;
b = a;
```

c)
```
a = temp;
int temp = b;
b = a;
```

d)
```
int temp = a;
b = a;
a = temp;
```

e)
```
int temp = a;
a = b;
b = temp;
```

f)
```
int temp = a;
b = temp;
a = b;
```

Answer e)

Question 6 What will be the value of p in terms of n and b after executing the following?

```
int p, n, b, i;
n = scanner.nextInt();
b = scanner.nextInt();
i = 1;
p = 1;
while (i < n)
{
    p = p * b;
    i = i + 1;
}
```

a) The value of p is b^{n-1}
b) The value of p is n^b
c) The value of p is b^n
d) The value of p is $(n-1)^b$
e) The value of p is $n^b - 1$

Answer a)

Question 7 What will be printed by the following?

```
String s = "1010110";
int n = 0, i = s.length() - 1, r = 1;
final int BASE = 2;

while (i >= 0)
{
    char c = s.charAt(i);
    if (c == '1')
    {
        n = n + r;
    }
    r = r * BASE;
    i = i - 1;
}

System.out.println(n);
```

a) 86
b) 1111
c) 241664
d) n
e) 1246

Answer a)

2 Short problems

Question 1 Consider the following class definitions:

```
class LightBulb {
    int id;
    LightBulb(int n) { id = n; }
    void shatters()
    {
        System.out.println(id + " is broken");
    }
}
class Box {
    LightBulb  a_light_bulb;
    void put(LightBulb b) { a_light_bulb = b; }
    void shake()
    {
        System.out.println("Shaking");
        a_light_bulb.shatters();
    }
}
```

Given these two classes, write a small code fragment that creates a box, and two light-bulbs, and then shatters one light-bulb, puts the second light-bulb in the box and then shakes the box. Also, write what would be printed by such fragment.

Code	Printout

`Box b;`
`LightBulb l1, l2;`
`b = new Box();`
`l1 = new LightBulb(1);`
`l2 = new LightBulb(2);`
`l1.shatters();`
`b.put(l2);`
`b.shake();`
	1 is broken
	Shaking
	2 is broken

Question 2 Consider the following code fragment:

```
String message = "secrets";
int index1 = 0, index2 = 0;
char letter1, letter2;

while (index1 < message.length())
{
    letter1 = message.charAt(index1);
    if (letter1 == 'e' || letter1 == 'c')
    {
        index2 = index1;
        while (index2 < index1 + 3 && index2 < message.length())
        {
            letter2 = message.charAt(index2);
            System.out.print(letter2);
            index2 = index2 + 1;
        }
    }
    else
    {
        System.out.print(letter1);
    }
    System.out.println();
    index1 = index1 + 1;
}
```

a) What will the above code print when executed?

```
s
ecr
cre
r
ets
t
s
```

b) What will the above code print when we remove the else block (including its print statement)?

```
ecr
cre
ets
```

Question 3 A *sequence of execution*, sometimes also called a *trace*, is a sequence of statements that describes the order of execution of one run of a program or fragment. For example, the trace for the code fragment:

```
System.out.print("Hello, "); // Stmt #1
System.out.print("Wor ");    // Stmt #2
System.out.print("ld! ");    // Stmt #3
```

is $(1, 2, 3)$. Consider the following program:

```
public class Tester {
    boolean isNegative(int y)
    {
        boolean b;                  // Stmt # 1
        b = false;                  // Stmt # 2
        if (y < 0) {
            b = true;               // Stmt # 3
        }
        return b;                   // Stmt # 4
    }
}
public class SimpleTest {
    public static void main(String[] args)
    {
        int x;                                          // Stmt # 5
        Scanner scanner = new Scanner(System.in);       // Stmt # 6
        System.out.println("Enter an integer");         // Stmt # 7
        x = scanner.nextInt();                          // Stmt # 8
        Tester t = new Tester();                        // Stmt # 9
        if (t.isNegative(x)) {
            System.out.println(x + " is negative.");    // Stmt # 10
        }
        else if (x == 0) {
            System.out.println(x + " is zero.");        // Stmt # 11
        }
        else {
            System.out.println(x + " is positive");     // Stmt # 12
        }
    }
}
```

a) Give one possible trace of execution of this program.

b) At the time in the execution when Statement 8 is executed, which variables (other than `args`) are in scope (that is, which variables are accessible at that point)?

c) At the time when Statement 2 is executed, which variables (other than `args`) are in scope?

(You can put your answers on the back of this page or the back of the previous page.)

a)

- If the user enters a positive number, the trace is: (5,6,7,8,9,1,2,4,12)
- If the user enters a negative number, the trace is: (5,6,7,8,9,1,2,3,4,10)
- If the user enters 0, the trace is: (5,6,7,8,9,1,2,4,11)

b) `x` and `scanner`

c) `y` and `b`

3 Programming problems

This section has three questions, but you have to answer only two of them. You must answer question 3, but you can choose whether to answer question 1 or question 2. State here which of the first two you answer: [ ]

Question 1 Two (positive) integers are *relatively prime* if their only common divisor (factor) is 1 (one). A *common divisor* (or factor) of two integers a and b is an integer d that divides both a and b, that is, the remainder of dividing a by d is 0 and the remainder of dividing b by d is also 0. For example, 15 and 28 are relatively prime, but 15 and 36 are not because 3 is a common factor.

Write a method that has two parameters `a` and `b`, and returns `true` if they are relatively prime, or `false` if they are not. Don't write an entire class; just the method is sufficient.

```
boolean areRelativePrimes(int a, int b)
{
    int smaller = a;
    if (b < a)
    {
        smaller = b;
    }

    int d = 2;

    while (d < smaller)
    {
        if (a % d == 0 && b % d == 0)
        {
            return false;
        }
        d = d + 1;
    }

    return true;
}
```

Question 2 Write a method to convert a given decimal number to binary. The method should accept a decimal number as a parameter (you can assume it is a positive integer,) and return a string of zeros and ones that represent the binary number. Don't write an entire class; just the method is sufficient.

```
String dec2bin(int decimal)
{
    String binary = "";
    if (decimal == 0)
    {
        binary = "0";
    }

    int quotient, remainder;

    quotient = decimal;
    while (quotient > 0)
    {
        remainder = quotient % 2;
        binary = remainder + binary;
        quotient = quotient / 2;
    }
    return binary;
}
```

Question 3 Write a class to represent the dashboard dials of a car. It must include:

- the current speed in kilometers per hour (0 - 160)
- the fuel guage in liters (0.0 - 50.0)
- a low-fuel warning light (true - false)
- the odometer in kilometers (0 - 9,999,999): this meassures the total distance traveled since the car was created.
  - this can never be reset, but if it reaches the upper limit, it cycles back to 0
- the trip odometer in kilometers (0 - 9,999,999): this meassures the total distance traveled since the odometer was last resetted.
  - this can be reset, and if it reaches the upper limit, it cycles back to 0

The values are not allowed to go out of range. When the class is instantiated as an object, all the instance data should be initialized appropriately... pretend that the car is fresh off the press. The class must have methods for accelerating and decelerating the car; increasing and decreasing the fuel level (the low-fuel warning light should turn on when the level reaches 10 percent of the tank's capacity); increasing (and rolling-over) both odometers; and resetting just the trip odometer. These are all mutators (setters.) The class must also have all the appropriate getter methods as well as a toString method.

You can assume that when a car is accelerated (or decelerated,) the only effect is that the current speed is incremented (or decremented) by a given amount. The same goes for the other methods: when increasing or decreasing the fuel, only the fuel and low-fuel warning are affected, and when increasing each odometer, only the odometers are affected.

There is no need to write a driver class with a main method.

```
class DashBoard
{
    double speed, fuel, tripDistance, totalDistance;
    boolean lowFuel;
    final double TANK_CAPACITY = 50.0;

    DashBoard()
    {
        speed   = 0.0;
        fuel    = 0.0;
        lowFuel = true;
        tripDistance = 0.0;
        totalistance = 0.0;
    }

    void accelerate(double amount)
    {
        speed = speed + amount;
        if (speed > 160.0)
        {
```

```
            speed = 160.0;
        }
    }
    void decelerate(double amount)
    {
        speed = speed - amount;
        if (speed < 0.0)
        {
            speed = 0.0;
        }
    }

    void increaseFuel(double amount)
    {
        fuel = fuel + amount;
        if (fuel > TANK_CAPACITY)
        {
            fuel = TANK_CAPACITY;
        }
        if (fuel <= 0.1 * TANK_CAPACITY)
        {
            lowFuel = true;
        }
        else
        {
            lowFuel = false;
        }
    }
    void decreaseFuel(double amount)
    {
        fuel = fuel - amount;
        if (fuel < 0.0)
        {
            fuel = 0.0;
        }
        if (fuel <= 0.1 * TANK_CAPACITY)
        {
            lowFuel = true;
        }
        else
        {
            lowFuel = false;
        }
    }

    void addDistance(double distance)
    {
```

```
            tripDistance = tripDistance + distance;
            if (tripDistance > 9999999.0)
            {
                tripDistance = tripDistance - 9999999.0;
            }
            totalDistance = totalDistance + distance;
            if (tripDistance > 9999999.0)
            {
                totalDistance = totalDistance - 9999999.0;
            }
        }

        void resetTripOdometer()
        {
            tripDistance = 0.0;
        }

        double getSpeed() { return speed; }
        double getFuel() { return fuel; }
        boolean lowFuel() { return lowFuel; }
        double getTripDistance() { return tripDistance; }
        double getTotalDistance() { return totalDistance; }

        public String toString()
        {
            String warning = "";
            if (lowFuel)
            {
                warning = "Warning: low fuel";
            }
            return "Current speed = " + speed
                +  " Fuel = " + fuel
                +  " " + warning
                +  " Odometer = " + totalDistance
                +  " Trip odometer = " + tripDistance;
        }
    }
```