THE UNIVERSITY OF WESTERN AUSTRALIA

Computer Science and Software Engineering

SEMESTER 1, 2015 EXAMINATIONS

CITS1001
Object-oriented Programming and Software Engineering

FAMILY NAME:_____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE:_____

This Paper Contains: **22** pages **(including title page)**
Time allowed: **2:10 hours (including reading time)**

INSTRUCTIONS:
Answer all questions. The paper contains nine questions, each worth ten marks.
Write your answers in the spaces provided on this question paper.
No other paper will be accepted for the submission of answers.
Do not write in this space.

				X	

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.

Supervisors Only - Student left at:

This page has been left intentionally blank

This page has been left intentionally blank

Class structure

1a) Write a Java class `State` to represent a state of Australia. The class should have

- three instance variables that capture a state's name, its time zone, and whether it is currently on Daylight Savings Time;
- a constructor that sets up each of these variables;
- accessor methods for each of these variables; and
- a mutator method for the DST variable. **(4 marks)**

1b) Write a Java class `Australia` that has

- one instance variable that can contain information about the six states of Australia;
- an accessor method that takes an integer and that returns the corresponding `State` object; and
- a method `timeDifference` that takes two states as arguments and that returns the current time difference between them, allowing for the possibility that one or both may be on DST. **(6 marks)**

Use this space and the following page for your answer to this question

```
public class State
{
    private String name;
    private int timeZone;
    private boolean DST;

    public State(String n, int tz, boolean d)
    {
        name = n;
        timeZone = tz;
        DST = d;
    }

    public String getName()
    {return name;}

    public int getTimeZone()
    {return timeZone;}

    public boolean getDST()
    {return DST;}

    public void setDST(boolean b)
    {DST = b;}
}


public class Australia
{
    private State[] states;

    public State getState(int k)
    {return states[k];}

    public int timeDifference(State x, State y)
    {
        int tx = x.getTimeZone();
        if (x.getDST()) tx++;
        int ty = y.getTimeZone();
        if (y.getDST()) ty++;
        return Math.abs(tx - ty);
    }
}
```

Numbers

2) Write the bodies of the three methods marked `TODO` below.

```
public class Circle
{

double cx, cy; // coordinates of the centre of this circle
double r;      // radius of this circle

public Circle(double cx, double cy, double r)
{
    if (r <= 0)
       throw new IllegalArgumentException("Negative radius");
    this.cx = cx;
    this.cy = cy;
    this.r  = r;
}

// returns the distance between points x1,y1 and x2,y2
public double distance(double x1, double y1, double x2, double y2)
{
    // TODO                                                    (3 marks)
}

// returns true iff point x,y is inside this circle
public boolean inside(double x, double y)
{
    // TODO                                                    (3 marks)
}

// returns true iff circle c is completely inside this circle
public boolean contains(Circle c)
{
    // TODO                                                    (4 marks)
}

}
```

Use this space and the following page for your answer to this question

```
// returns the distance between points x1,y1 and x2,y2
public double distance(double x1, double y1, double x2, double y2)
{
    return Math.sqrt(Math.pow(x1 - x2, 2) + Math.pow(y1 - y2, 2));
}


// returns true iff point x,y is inside this circle
public boolean inside(double x, double y)
{
    return distance(x, y, cx, cy) <= r;
}


// returns true iff circle c is completely inside this circle
public boolean contains(Circle c)
{
    return inside(c.cx, c.cy) &&
           distance(cx, cy, c.cx, c.cy) + c.r <= r;
}
```

Conditionals

3a) Using only conditionals and relational operators, write the body of the method `middle` that takes three numbers and returns the middle number, by size. For example, `middle(4, 7, -7)` returns `4`, and `middle(4, 7, 7)` returns `7`. **(5 marks)**

```
// middle returns the median of x, y, and z
public int middle(int x, int y, int z)
```

3b) Using only conditionals and relational or logical (Boolean) operators, write the body of the method `one` that takes three Boolean values and returns `true` if and only if exactly one of its argument is `true`. **(5 marks)**

Use this space and the following page for your answer to this question

```
public int middle(int x, int y, int z)
    {
        if (x <= y)
           if (y <= z) return y;
           else
           if (x <= z) return z;
           else        return x;
        else
           if (x <= z) return x;
           else
           if (y <= z) return z;
           else        return y;
    }


public boolean one(boolean x, boolean y, boolean z)
    {
        if (x) return !y && !z;
        else   return y != z;
    }
```

Arrays

4a) Write the body of the method `ison` that returns `true` if and only if the number `x` is in the array `a`. For example, `ison(3, {5,4,3,2})` returns `true`, but `ison(3, {5,4,2})` returns `false`. **(3 marks)**

```
// returns true iff x is on a
public static boolean ison(int x, int[] a)
```

4b) Use `ison` to write the body of the method `sameElements` that returns `true` if and only if the arrays `a` and `b` contain the same elements, although possibly in different orders. You may assume that neither array contains any duplicated elements. For example, `sameElements({3,1}, {1,3})` returns `true`, but `sameElements({3,1}, {2,3})` and `sameElements({3}, {3,2}` both return `false`. **(5 marks)**

```
// returns true iff a and b contain the same elements
public static boolean sameElements(int[] a, int[] b)
```

4c) Does your definition of `sameElements` work correctly if the arrays are allowed to contain duplicates? Justify your answer. **(2 marks)**

Use this space and the following page for your answer to this question

```
// returns true iff x is on a
public static boolean ison(int x, int[] a)
    {
        for (int y : a)
            if (x == y)
               return true;
        return false;
    }
```

```
// returns true iff a and b contain the same elements
public static boolean sameElements(int[] a, int[] b)
    {
        for (int x : a)
            if (!ison(x, b))
               return false;
        return a.length == b.length;
    }
```

It won't work, e.g. it returns `true` for `sameElements({3,1,1}, {1,3,3})`.

Display

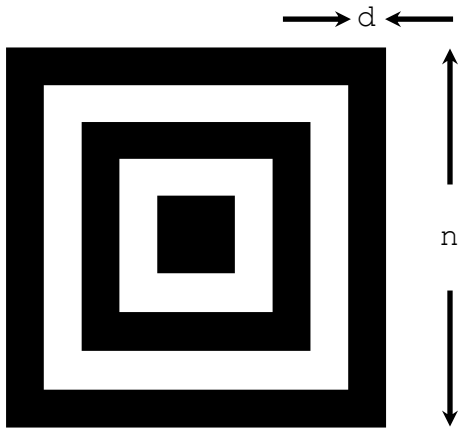
For the following questions, you may assume that both `java.awt.Color` and `SimpleCanvas` have been imported.

5a) Write the body of the method `flip` that takes a colour as argument: if the argument is white, it returns black; if the argument is black, it returns white; if the argument is anything else, it throws an `IllegalArgumentException`. **(3 marks)**

```
// returns the opposite colour to col
public Color flip(Color col)
```

5b) Use `flip` and the `SimpleCanvas` method `drawSq` below to write the body of the method `nestSquares` that takes numbers `n` and `d`, and that draws nested squares as shown below, where `n` is the side of the largest square, and `d` is the thickness of each square in the nest. **(7 marks)**

```
// drawSq(x, y, s, c) draws a square of side s and
// colour c on sc, with its top-left corner at x,y
public void drawSq(int x, int y, int s, Color c)
```




```
// draws the nest of squares with size n and thickness d
public void nestSquares(int n, int d)
```

Use this space and the following page for your answer to this question

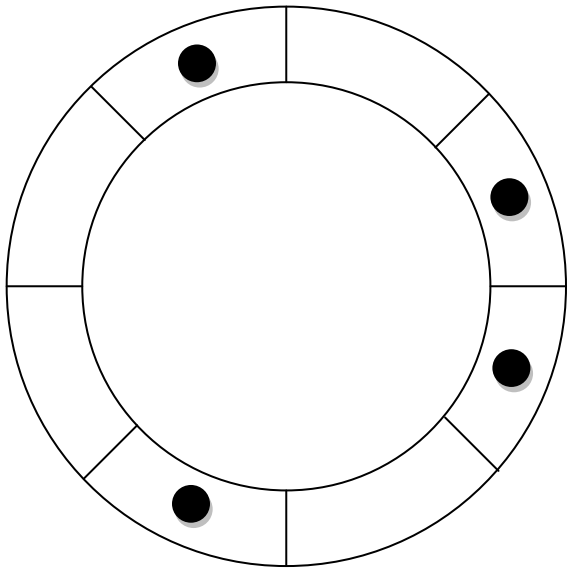
```
// returns the opposite colour to col
public Color flip(Color col)
    {
        if (col == Color.black) return Color.white;
        else
        if (col == Color.white) return Color.black;
        else
        throw new IllegalArgumentException("Bad colour");
    }



// draws the nest of squares with side n and thickness d
public void nestSquares(int n, int d)
    {
        SimpleCanvas sc = new SimpleCanvas();
        java.awt.Color col = Color.black;
        for (int i = 0; i < n / 2; i = i + d)
            {
                sc.drawSq(i, i, n - 2 * i, col);
                col = flip(col);
            }
    }
```

Cellular automata

6) A one-dimensional cellular automaton is a circular strip of cells, each of which is either on or off at any given moment.



An automaton with *n* cells is indexed from *0* to *n–1*, and these two cells are neighbours in the circle. A particular automaton defines a rule by which the status of each cell in Generation *k* is determined from the status of the cell and its immediate neighbours in Generation *k–1*. One famous rule is Rule 110, defined as:

111	110	101	100	011	010	001	000
0	1	1	0	1	1	1	0

For example, the seventh cell of the table says that if Cell *j* is off (i.e. 0 = false), its left neighbour is off, and its right neighbour is on, then Cell *j* is set on in the next generation.

Write the body of the method `nextGeneration` that takes the state of an automaton in one generation and returns the state of the automaton in the next generation, derived using Rule 110. **(10 marks)**

```
// updates map to the next generation
public void nextGeneration(boolean[] map)
```

Use this space and the following page for your answer to this question

```
// updates map to the next generation
public void nextGeneration(boolean[] map)
    {
        int n = map.length;
        boolean[] nextmap = new boolean[n];
        for (int i = 0; i < n; i++)
        {
            int l = (i - 1 + n) % n;
            int r = (i + 1)     % n;
            if (map[r] && (map[l] || !map[i]))
               nextmap[i] = !map[i];
            else
               nextmap[i] =  map[i];
        }
        map = nextmap;
    }
```

Sorting

7) "Counting sort" processes an array `a` of integers by counting how many of each number occurs in `a`, then assigning the elements of `a` to the result array `b` using the counts to determine their locations. It is implemented by the static method `countingSort`.

```
// returns a sorted copy of a, assuming that it contains
// only integers in the range 0 .. k-1
public static int[] countingSort(int[] a, int k)
{
    int[] counts = new int[k];
    for (int x : a)
        counts[x]++;
    int total = 0;
    for (int i = 0; i < k; i++)
    {
        int oldCount = counts[i];
        counts[i] = total;
        total += oldCount;
    }
    int[] result = new int[a.length];
    for (int x : a)
    {
        result[counts[x]] = x;
        counts[x]++;
    }
    return result;
}
```

a) Show how the application `countingSort({3,7,1,3,8,2,1}, 10)` is processed. In particular, show the state of the array `counts` at the end of the first loop, and at the end of the second loop; and how `counts` and `result` change during the third loop. **(7 marks)**

b) Comment on the performance of `countingSort`, as formally as you can. For what data will the algorithm be especially fast? **(3 marks)**

Use this space and the following page for your answer to this question

a = {3, 7, 1, 3, 8, 2, 1}
k = 10

counts after the first loop: [0, 2, 1, 2, 0, 0, 0, 1, 1, 0]

counts after the second loop: [0, 0, 2, 3, 5, 5, 5, 5, 6, 7]

result = [0, 0, 0, 0, 0, 0, 0], counts = [0, 0, 2, 3, 5, 5, 5, 5, 6, 7]
result = [0, 0, 0, 3, 0, 0, 0], counts = [0, 0, 2, 4, 5, 5, 5, 5, 6, 7]
result = [0, 0, 0, 3, 0, 7, 0], counts = [0, 0, 2, 4, 5, 5, 5, 6, 6, 7]
result = [1, 0, 0, 3, 0, 7, 0], counts = [0, 1, 2, 4, 5, 5, 5, 6, 6, 7]
result = [1, 0, 0, 3, 3, 7, 0], counts = [0, 1, 2, 5, 5, 5, 5, 6, 6, 7]
result = [1, 0, 0, 3, 3, 7, 8], counts = [0, 1, 2, 5, 5, 5, 5, 6, 7, 7]
result = [1, 0, 2, 3, 3, 7, 8], counts = [0, 1, 3, 5, 5, 5, 5, 6, 7, 7]
result = [1, 1, 2, 3, 3, 7, 8], counts = [0, 2, 3, 5, 5, 5, 5, 6, 7, 7]

The first loop takes time proportional to n = a.length
The second loop takes time proportional to k
The third loop takes time proportional to n
So it is O(n + k) in total

It will be very fast if k–n is small

Recursion

8a) What is the role of the *base cases* in a recursive Java method? **(2 marks)**

8b) The greatest common divisor of two positive integers is the largest integer that divides exactly into both numbers with no remainder. Euclid's algorithm for calculating the greatest common divisor of two positive integers uses two rules:

gcd(x, 0) = x
gcd(x, y) = gcd(y, remainder of x/y), if y > 0

Write the body of the method `gcd` that calculates greatest common divisors using this algorithm. **(5 marks)**

```
// returns the greatest common divisor of x and y
public static int gcd(int x, int y)
```

8c) Show the sequence of method calls that your definition generates for the invocation `gcd(22,24)`. **(3 marks)**

Use this space and the following page for your answer to this question

They tell the method when to stop; they return a result directly.

```
// returns the greatest common divisor of x and y
public static int gcd(int x, int y)
    {
        if (y == 0) return x;
        else        return gcd(y, x % y);
    }
```

```
   gcd(22, 24)
→  gcd(24, 22)
→  gcd(22,  2)
→  gcd( 2,  0)
→  2
```

Exceptions

9a) What are exceptions used for in Java? **(2 marks)**

9b) What is the difference between a *checked exception* and an *unchecked exception*? **(2 marks)**

9c) Give two examples of situations where a checked exception would be used, and two where an unchecked exception would be used. **(2 marks)**

9d) Give the outline of a method `M1` that causes a checked exception, and the outline of a method `M2` that illustrates how `M1` must be called. **(4 marks)**

Use this space and the following page for your answer to this question

Exceptions are used to signify errors that prevent code from executing normally

An unchecked exception causes the program to crash
A checked exception can be caught by another method, and the program can recover

Checked exceptions: file not found, URL unavailable
Unchecked exceptions: array index out of range, dereferencing a null pointer

```
private void M1() throws FileNotFoundException
{
    if (condition) {execute}
    else throw new FileNotFoundException();
}


private type M2()
{
    try   {code that invokes M1()}
    catch {code that tries to recover}
}
```

END OF PAPER