CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DAT 050	Objektorienterad Programmering			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
DAT050-0009-MRX		26/10 - 23	7	5*

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	12	
2	X	~~8~~ 10	
3	X	15	
4	X	2 ~~4~~ 3	
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		60 ***	

[redacted]

1

A) • Oftast vill man att sin kod ska vara modulär. med detta menas hög kohesion och låg koppling.

med hög kohesion menas att en kod (ex metod) har en begränsad uppgift den utför och är "mindre allmän". Låg kohesion menar på en mera allmän kod. exempelvis kan man ha en metod med hög kohesion som tar ut just 5:e elementet i en ArrayList bara om den innehåller Pelle objekt.

• Koppling kan man se som återanvändbarheten av kod.

hög koppling menar att en kod kan återanvändas i andra delar av ett program, medans låg koppling menar för motsatsen.

exempelvis kan en public static generic metod ge en hög koppling och återanvändbarhet. 3p

B) Immutable (icke-muterbara) klasser har alltid samma värde efter att ett objekt skapas, och kan ej manipuleras via ex setters. Istället om ett objekt ska få ett nytt värde så skapar man en ny referens till ett nytt objekt med "new" nyckelordet, och till delar den till objektet som ska "muteras". 3p

C) Specifikation är något vi ofta ser i interfaces. Dem specifierar vilka metoder som bör finnas, men ej hur dem implementeras. ex: void moveX();

Medans en specifikation medför att kod är implementerad för specifikationen. ex:
```
public void moveX() {
    this.x++;
}
```
3p

Både specifikations arv och implem. arv är möjliga i senare Java versioner.

D) En klassvariabel deklareras med "static" keyworset och kan ej nås med "this". Den delar minne och värde med alla objekt av klassen, medans instansvariabler är unika för varje instans. Ex:
```
public static int a = 5;
public int b = 10;
:
Classname c1 = new Classname(); , samma för c2...
c1.a = 15;  och c1.b = 70.
nu är c2.a 15  och c2.b är 10.
```
3p

1

//Det står ej explicit i uppgiften men jag antar //att vi bara ska skriva en ny metod som använder en map.

```
//klass var:
private static Map<BigInteger, Integer> map = new HashMap<>();

public static int getCollatzIterationsWithMap(BigInteger v){
    Integer num = map.get(v); BigInteger org = new BigInteger(v);
    int count = 0;
    if(num == null){
        while(!v.equals(BigInteger.ONE)){
            count++;
            v = nextCollatz(v);
        }
        map.put(org, new Integer(count));
        return count;
    }
    else { return count + num;} //Auto-unboxing of Integer to int.
}                                (for num)
```

Bra!

~~- Medelvärde och så~~ -3

~~7p~~

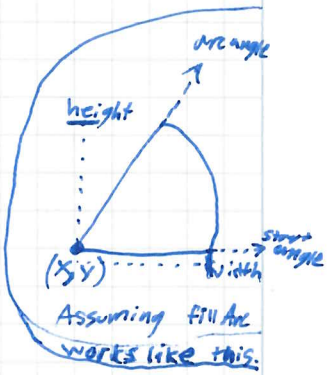
10p

1

```
Public Class PiePanel extends JPanel {
    Private double[] data;
    Private double sum = 0,0; Private int count = 0; - Används ej.
    Public PiePanel(double[] data) {
        this.data = data.;clone();
        for(double d : data) {
            sum += d; count++;
        }
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int w = getWidth();
        int h = getHeight();
        //Draw circle (not filled)
        g.drawOval(0, 0, w, h);
        double prevAngle = 0;

        for (double d : this.data) { //draw arcs and lines.
            double angle = (d/this.sum) * 360; //arc angle
            g.setColor((color.getHSBColor((float) angle, 0.9f, 0.9f));
            g.fillArc((int) w/2,(int) h/2, (int) w/2, (int)h/2, prevAngle, angle);
            prevAngle += angle;
            //Draw lines:
            double lineX = Math.cos(prevAngle + angle) * (w/2) + (w/2);
            double lineY = -Math.sin(prevAngle + angle) * (h/2) + (h/2);
            g.setColor(color.BLACK);
            // Don't know the exact name of following method, but a drawLine exists.
            g.drawLine((int) w/2, (int) h/2 , lineX, lineY);
        }
    }

    //assume main from description is here.

} //end of class.
```

g.setColor(color.Black);

//total angle



Bra!

15p

1

```
Public Class MemoryModel {
Private int[][] cardField; Private int width; Private int height;
private List<Integer> cardList;
Private int pairwiseGuesses = 0; private int score = 0;
public MemoryModel(int width, int height) {
    this.width = width; this.height = height.
    int numCards = width.height;
    this.cardList = new ArrayList<>();
    for(int i=0; i<numcards; i++){
        this.cardList.add(i);
        this.cardList.add(i);
    }
    Collections.shuffle(this.cardList);
    this.cardField = new int[width][height];
    int count = 0;
    for(int row = 0; row < height; row++) {
        for(int col =0; col < width; col++) {
            this.cardField[row][col] = this.cardList.get(count);
            count++;
        }
    }
}

Public double getScore() {
    return (double) this.score / this.pairwiseGuesses.
}

public int getHeight() { return this.height }
public int getWidth() { return this.width }

public final boolean isGameOver() {
    if(this.score == ((this.width * this.height)/2)) {
        return true;
    }
    return false;
}

Public int[][] getField() { return this.cardField } //Should do a copy...
```

24p!

constructor

getScore

getW
getH

isGameOver

//Next Page ▽

```
public int getValue(int x, int y) {
    return this.cardField[x][y];
}

public boolean guess(int x1, int y1, int x2, int y2) {
    if(this.cardField[x1][y1] == this.cardField[x2][y2]) {
        this.pairwiseGuesses++;
        this.score++;
        return true;
    }
    this.pairwiseGuesses++;
    return false;
}

Public void restart() {
    this.score = 0;
    this.pairwiseGuesses = 0;
    Collections.shuffle(this.cardList);
    int count = 0;
    for(int row = 0; row < this.height; row++) {
        for(int col = 0; col < this.with; col++) {
            this.cardField[row][col] = this.cardList.get(count);
            count++;
        }
    }
}

} //end of Memory Model.
```

```
Public Class MemoryController extends JPanel {
    Private MemoryModel m;
    Private JButton[][] buttons;
    Private boolean oneClicked = false;
    Public MemoryController(MemoryModel m) {
        this.m = m;
        JFrame f = new JFrame("Memory");
        f.setDefaultCloseOperation(JFrame.EXITON_CLOSE);
        f.add(this)
        this.setLayout(new BorderLayout);
        // osäker på syntax för att lägga till i panelerna., men
        // vill ha grid med knappar i Center och resten South...
        JPanel buttonGrid = new JPanel();
        buttonGrid.setLayout(new GridLayout(m.getWidth(), m.getHeight()));
        this.add(buttonGrid, JPanel_CENTER); //antar detta syntax.
        JPanel bottomGrid = new JPanel();
        bottomGrid.setLayout(new GridLayout(2, 1));
        JLabel score = new JLabel("score: " + m.getScore());
        JButton restart = new JButton("Restart");
        restart.addActionListener((e) -> { m.restart(); });
        bottomGrid.add(score);
        bottomGrid.add(restart);
        this.add(bottomGrid, JPanel_SOUTH);
        this.buttons = new JButton[m.getWidth()][m.getHeight()]
        fillButtons(this.buttons); f.pack(). f.setVisible(true);
    }
}
```

buttonGrid.

lower grid score+restart

//Next Page ▽ (fillButtons finns där)

```
private static void fillButtons (JButton[][] but) {
    for(int row = 0; row < m.getHeight(); row++) {
        for(int col = 0; col < m.getWidth(); col++) {
            but[row][col] = new JButton(" ");
            but[row][col].addActionListener((e) -> {
                if (this.oneClicked == false) {
                    but[row][col].setText(m.getField[row][col]);
                    this.oneClicked = true;
                }
                else if(this.oneClicked == true) {
                    if ( but[row][col].getActionCommand().equals(första knapp ens värde...))
                    { but[row][col].setBackground(Color.GREEN);
                      but[row][col].setEnabled(false);
                      but[row][col].setText(m.getField[row][col]);
                    this.m.guess( row, col , last_pressed_x, last_pressed_y);
                }
            }
        }
    }
}

// har slut på tenta-tid, men borde ha sparat
// den senast tryckta knappen som instansvariabel
// så att nästa kan jämföras, och kolla isGameOver
// för att se om spelet har slut
```