Lösningsförslag till tentamen 140423

Uppgift 1

a) Vad kompilatorn klagar på är att metoden Math.sqrt() inte finns! Detta kan ju verka något mystiskt eftersom vi alla vet att det i klassen Math finns en klassmetod sqrt(). Alla vet också att klassen Math finns i paketet java.lang och att detta paket inte behöver importeras. Så varför blir det då fel?

Jo, problemet är att också den egna klassen heter Math!! Vid namngivning i Java gäller närhetsregeln, därför avser Math i anropet Math.sqrt(tal) den egna klassen Math och inte java.lang.Math. Kompilatorn har givetvis rätt, i den egna klassen finns ingen metod sqrt definierad.

För att rätta till felet skall vi således specificera att det är klassen java.lang.Math vi avser:

```
import javax.swing.*;
public class Math {
    public static void main (String[] arg) {
        String indata = JOptionPane.showInputDialog("Ange ett tal: ");
        double tal = Double.parseDouble(indata);
        double root = java.lang.Math.sqrt(tal);
        JOptionPane.showMessageDialog(null, "Roten ur talet " + tal + " är " + root);
    } //main

} //Math
```

Vi skulle också kunna ändra namnet på den egna klassen:

```
import javax.swing.*;
public class AnOtherMath {
    public static void main (String[] arg) {
        String indata = JOptionPane.showInputDialog("Ange ett tal: ");
        double tal = Double.parseDouble(indata);
        double root = Math.sqrt(tal);
        JOptionPane.showMessageDialog(null, "Roten ur talet " + tal + " är " + root);
    } //main
} //AnOtherMath
```

Uppgiften var dock formulerad på så sätt att förändringarna skulle göras i filen Main.java, dvs. den egna klassen får inte döpas om.

b) Den styrande variabeln index i **for**-satsen löper till och värdet vekt.length -1, vilket är index för det sista elementet i fältet vekt. Inuti **for**-satsen refereras dock till fältelement vekt[index+1]. **for**-satsen löper helt enkelt *ett varv för mycket*. En korrekt version av metoden skall ha följande utseende:

```
public static boolean  sorted(int[] vekt){
    boolean  okey = true ;
    for  (int  index = 0; index < vekt.length - 1; index = index + 1)
        if (vekt[index] > vekt[index+1])
            okey = false ;
    return  okey;
} //sorted
```

c) Metoden nextInt är en instansmetod och inte en statisk metod, därför är anropet

```
Random.nextInt(10)
```

felaktigt. Vi måste skapa en instans av Random och anropa denna instans:

```
public static void main (String[]  arg) {
    Random rand = new Random();
    for (int i = 1; i <= 100; i = i + 1) {
        int val = rand.nextInt(10) + 1;
        System.out.println("Random number between 1 and 10 is " + val);
    }
}//main
```

Uppgift 2

Lösning med användning av dialogrutor:

```
public static void main(String[] args) {
    while (true) {
        String indata = JOptionPane.showInputDialog("Ange lånat belopp, antal år och ränta : ");
        if (indata == null)
            break;
        Scanner sc = new Scanner(indata);
        double loan = sc.nextDouble();
        int years = sc.nextInt();
        double rate = sc.nextDouble();
        if (loan <= 0 || years <= 0 || rate <= 0)
            JOptionPane.showMessageDialog(null, "Felaktig indata! Försök igen!");
        else
            JOptionPane.showMessageDialog(null, "Månatlig kostnad blir: " +
                                                String.format("%.0f kronor." , calculatePayment(loan, rate, years)));
    }
} //main
```

Lösning med användning av System.in och System.out och String.format:

```
import java.util.*;
public class Mortgage {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        while (true) {
            System.out.print("Ange lånat belopp, antal år och ränta : ");
            if (!console.hasNext())
                break;
            double loan = console.nextDouble();
            int years = console.nextInt();
            double rate = console.nextDouble();
            if (loan <= 0 || years <= 0 || rate <= 0)
                System.out.println( "Felaktig indata! Försök igen!");
            else
            System.out.println("Månatlig kostnad blir: " + (int) calculatePayment(loan, rate, years) + " kronor.");
        }
    } //main

    private static double calculatePayment(double loan, double rate, int years) {
        int n = 12 * years;
        double c = rate / 12.0 / 100.0;
        return loan * c * Math.pow(1 + c, n) / (Math.pow(1 + c, n) - 1);
    }//calculatePayment
} //Mortgage
```

Uppgift 3

```
public static boolean verifyCreditCardChecksum(int[] digits) {
    int sum = 0;
    for (int i = 0; i < digits.length; i = i + 1)  {
        if (i % 2 == 0) {
            int value = digits[i] * 2;
            sum = sum + value / 10 + value % 10;
        }
        else {
            sum = sum + digits[i];
        }
    }
    return sum % 10 == 0;
}// verifyCreditCardChecksum
```

Alternativ lösning med användning av hjälpmetoder:

```
public static boolean verifyCreditCardChecksum(int[] digits) {
    int[] doubledValues = doubleEvenIndices(digits);
    int[] summedDigits = sumDigits(doubledValues);
    return sum(summedDigits) % 10 == 0;
}// verifyCreditCardChecksum

private static int[] doubleEvenIndices(int[] digits)  {
    int[] doubledValues = new int[digits.length];
    for (int i = 0; i < digits.length; i = i + 1)  {
        if (i % 2 == 0)
            doubledValues[i] = digits[i] * 2;
        else
            doubledValues[i] = digits[i];
    }
    return doubledValues;
}// doubleEvenIndices

private static int[] sumDigits(int[] digits) {
    int[] summedDigits = new int[digits.length];
    for (int i = 0; i < digits.length; i = i + 1) {
        summedDigits[i] = digits[i];
    }
    return summedDigits;
}// sumDigits

private static int sum(int[] array) {
    int accumulate = 0;
    for (int i = 0; i < array.length; i++) {
        accumulate += array[i];
    }
    return accumulate;
}//sum
```

Uppgift 4

```
public class Uppgift4 {
  public static int[][][] fourCopies(int[][][] samples) {
    int[][][] newSamples = new int[samples.length][samples[0].length][3];
    for (int row = 0; row < samples.length/2; row++) {
      for (int col = 0; col < samples[row].length/2; col++) {
        for (int i = 0; i < 3; i = i + 1) {
            int value = mean(samples, row, col, i);
            newSamples[row][col][i] = value;
            newSamples[row+samples.length/2][col][i] = value;
            newSamples[row][col + samples[row].length/2][i] = value;
            newSamples[row + samples.length/2][col + samples[row].length/2][i] = value;
        }
      }
    }
    return newSamples;
  }//fourCopies

  public static int mean(int[][][] samples, int row, int col, int color) {
     int sum = samples[2*row][2*col][color] + samples[2*row][2*col + 1][color]
                + samples[2*row + 1][2*col][color] + samples[2*row + 1][2*col + 1][color];
     return (int) (sum / 4.0);
  }//mean
}//Uppgift4
```

Uppgift 5

a)

```
import java.util.Random;
public class LCRDice {
    private final Random rand = new Random();
    private final String[] symbols = {"L", "C", "R", ".", ".", "."};
    public String roll() {
        int i = rand.nextInt(5);
        return symbols[i];
    }
}
```

b)

```
    private Player getPlayerLeft() {
        int i = players.indexOf(actual);
        return players.get((i + 1) % players.size());
    }

    private Player getPlayerRight() {
        int i = players.indexOf(actual);
        if (i == 0) {
            i = players.size();
        }
        return players.get(i - 1);
    }

    private int getNRolls() {
        int nChip = actual.getNChips();
        if (nChip >= 3) {
            return 3;
        } else {
            return nChip;
        }
    }
```

c)

```
    public boolean gameOver() {
        int count = 0;
        for (Player p : players) {
            if (p.getNChips() > 0) {
                count++;
            }
        }
        return count == 1;
    }
```

d)

```
public void roll() {
    result.clear();
    for (int i = 0; i < getNRolls(); i++) {
        result.add(dice.roll());
    }
    for (String s : result) {
        switch (s) {
            case "L":
                actual.removeChip();
                getPlayerLeft().addChip();
                break;
            case "C":
                actual.removeChip();
                break;
            case "R":
                actual.removeChip();
                getPlayerRight().addChip();
                break;
            case ".": // Do nothing
                break;
        }
    }
    actual = getPlayerLeft();
}//roll
```

e)

```
public static void main(String[] args) {
    boolean done = false;
    List<Player> players = Arrays.asList(new Player("Pelle", 3), new Player("Fia", 3), new Player("Sven", 3));
    LCRGame lcr = new LCRGame(players, new LCRDice());
    lcr.start();
    System.out.println("LCR started");
    System.out.print("Players are ");
    dump(lcr);
    Scanner s = new Scanner(System.in);
    while (!done) {
        System.out.println("Player is " + lcr.getActualPlayer());
        System.out.print("> ");
        String cmd = s.nextLine();
        switch (cmd) {
            case "r":
                lcr.roll();
                dump(lcr);
                if (lcr.gameOver()){
                    done = true;
                }
                break;
            case "q":
                done = true;
                break;
            default:
                System.out.println("?");
        }
    }
    if (lcr.gameOver()) {
        System.out.println("Game over! Winner is " + lcr.getActualPlayer());
    } else {
        dump(lcr);
        System.out.println("Game aborted");
    }
}
```

Uppgift 6

```
import java.awt.GridLayout;
import javax.swing.JFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JLabel;
public class MainFrame extends JFrame implements ActionListener {
    private final JButton start = new JButton("Start");
    private final JButton stop = new JButton("Stop");
    private final JLabel result = new JLabel("Result");

    public MainFrame() {
        this.setLayout(new GridLayout(3, 1));
        this.setTitle("Reaction tester");
        this.add(start);
        this.add(stop);
        stop.setEnabled(false);
        this.add(result);
        start.addActionListener(this);
        stop.addActionListener(this);
        this.setSize(250, 150);
    }

    private long time;
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == start) {
            start.setEnabled(false);
            stop.setEnabled(true);
            time = System.currentTimeMillis();

        }else if (e.getSource() == stop){
            start.setEnabled(true);
            stop.setEnabled(false);
            time = System.currentTimeMillis() - time;
            result.setText("Result = "  + String.valueOf(time) + " : ms");
        }
    }
    public static void main(String[] args) {
        new MainFrame().setVisible(true);
    }
}
```