Lösningsförslag till tentamen 131220

Uppgift 1

a) Första felet beror på att namnet på klassen är felstavat i konstruktorn! Således har vi ingen konstruktor utan en metod med namnet Trubel. En metod måste specificera typen av på värdet metoden returnerar, just i detta fallet returneras inget värde varför returtypen skall vara **void** (om det hade varit en metod med namnet Trubel som vi verkligen skulle vilja haft). Dock är det en konstruktor vi vill ha (annars skulle aldrig instansvariabeln kunna ges ett värde). Alltså skall Trubel på rad 3 bytas mot Trubbel.

Avsikten med metoden getNumber är att returnera värdet på instansvariabeln, men denna heter aNumber och inte n. Alltså skall n på rad 7 bytas mot aNumber.

```
public class Trubbel{
    private int aNumber;
    public Trubbel(int n){
        aNumber = n;
    }
    public int getNumber(){
        return aNumber;
    }
}
```

b) Felet beror på villkoret wordArray[i] == word i **if**-satsen!

Här jämförs två referenser till objekt av typen String. Avsikten är att villkoret skall bli **true** om objekten refererar till objekt som har samma värde. Men likhetsoperatorn == ger **true** om och endast om referenserna refererar till samma objekt.

För att få det önskade resultatet skall objekten jämföras med equals-metoden.

```
public static int countOccurrences(String[] wordArray, String word) {
    int result = 0;
    for (int i = 0; i < wordArray.length; i = i + 1) {
        if (wordArray[i].equals(word)) {
            result = result + 1;
        }
    }
    return result;

}
```

Uppgift 2

Med användning av dialogrutor:

```
import javax.swing.*;
import java.util.*;
public class Speed {
    public static void main(String[] args) {
        while(true) {
            String indata = JOptionPane.showInputDialog("Ange avverkad sträcka i meter" +
                                                         " och använd tid i sekunder: ");
            if (indata == null)
                break;
            Scanner sc = new Scanner(indata);
            double sträcka = sc.nextDouble();
            double tid = sc.nextDouble();
            if (sträcka <= 0 || tid <= 0 ) {
                JOptionPane.showMessageDialog(null, "FELAKTIG INDATA!");
            }
            else {
                double speedMpS = sträcka / tid;
                double speedKpT = toKmPerHours(speedMpS);
                JOptionPane.showMessageDialog(null, "Medelhastigheten är " + String.format("%.2f",speedMpS)
                                                  + " meter per sekund, \nvilket motsvarar "
                                                  + String.format("%.2f", speedKpT) + " kilometer per timma.");
            }
        }
    }//main
    public static double toKmPerHours(double mPerS) {
        return mPerS * 60*60/1000;
    }// toKmPerHours
}//Speed
```

Med användning av System.in och System.out:

```
import java.util.*;
public class Speed {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while(true) {
            System.out.print("Ange avverkad sträcka i meter och använd tid i sekunder: ");
            if (!sc.hasNext())
                break;
            double sträcka = sc.nextDouble();
            double tid = sc.nextDouble();
            if (sträcka <= 0 || tid <= 0 ) {
                System.out.println("FELAKTIG INDATA!");
            }
            else {
                double speedMpS = sträcka / tid;
                double speedKpT = toKmPerHours(speedMpS);
                System.out.println("Medelhastigheten är " + String.format("%.2f",speedMpS) + " meter per sekund");
                System.out.println("vilket motsvarar " + String.format("%.2f", speedKpT) + " kilometer per timma.");
            }
        }
    }//main
    public static double toKmPerHours(double mPerS) {
        return mPerS * 60*60/1000;
    }// toKmPerHours
}//Speed
```

Uppgift 3

```
public int[] trimSilenceFromFront(int[] samples) {
    int nrOfZeros = 0;
    boolean  finished = false;
    while ( nrOfZeros < samples.length && !finished) {
        if (samples[nrOfZeros] == 0)
            nrOfZeros = nrOfZeros + 1;
        else
         finished = true;
    }
    int[] newSamples = new int[samples.length - nrOfZeros];
    for (int i = nrOfZeros; i < samples.length; i = i  + 1) {
            newSamples[i - nrOfZeros] = samples[i];
    }
    return newSamples;
}// trimSilenceFromFront
```

Uppgift 4

```
public static int[][][] hideMessage(int[][][] sample, String msg) {
    int[][][] newSample = new int[sample.length][sample[0].length][3];
    int nrOfLetters = 0;
    for (int x = 0; x < newSample.length; x = x + 1) {
        for (int y = 0; y < newSample[x].length; y = y + 1) {
            if (nrOfLetters < msg.length()) {
                char ch = msg.charAt(nrOfLetters);
                nrOfLetters = nrOfLetters + 1;
                int ascii = (int) ch;
                int r = ascii / 100;
                int g = ascii / 10 % 10;
                int b = ascii % 10;
                newSample[x][y][0] = 10 * (sample[x][y][0] / 10) + r;
                newSample[x][y][1] = 10 * (sample[x][y][1] / 10) + g;
                newSample[x][y][2] = 10 * (sample[x][y][2] / 10) + b;
            }
            else {
                for (int i = 0; i < 3; i++)
                    newSample[x][y][i] = sample[x][y][i];
            }
        }
    }
    return newSample;
}//hideMessage
```

Uppgift 5

a)

```
import java.util.Random;
public class Dice {
    private final int faces;
    private final Random random = new Random();
    private int result;
    private int total;

    public Dice(int faces) {
        this.faces = faces;
    }

    public int roll() {
        result = random.nextInt(faces) + 1;
        if (result > 1) {
            total += result;
        } else {
            total = 0;
        }
        return result;
    }

    public int getTotal() {
        return total;
    }

    public int getLastResult() {
        if (result > 0) {
            return result;
        } else {
            throw new IllegalStateException("Must roll dice first");
        }
    }

    public void clear() {
        total = 0;
    }
}
```

b)

```
import java.util.List;
import java.util.Random;
public class Pig {
    public static final int FACES = 6;
    private final List<Player> players;
    private Player actualPlayer;
    //private final Dice dice = new Dice(FACES);
    private final PigDice dice = new PigDice(FACES);

    public Pig(List<Player> players) {
        this.players = players;
    }

    public int roll() {
        return dice.roll();
    }

    public Player getActualPlayer() {
        return actualPlayer;
    }

    public List<Player> getPlayers() {
        return players;
    }

    public void start() {
        int playerNo = new Random().nextInt(players.size());
        this.actualPlayer = players.get(playerNo);
    }

    public void next() {
        dice.clear();
        int i = players.indexOf(actualPlayer);
        actualPlayer = players.get((i + 1) % players.size());
    }

    public void setTotal() {
        actualPlayer.addPoints(dice.getTotal());
    }

    public boolean gameOver() {
        return actualPlayer.getTotal() >= PigOptions.getPointsToWin();
    }
}
```

c)

```
import java.util.Scanner;
public class CommandLinePig {
    public static void main(String[] args) {
        boolean done = false;
        Pig pig = new Pig(PigOptions.getPlayers());
        pig.start();
        System.out.println("Pig started");
        System.out.print("Players are ");
        dump(pig);
        Scanner s = new Scanner(System.in);
        System.out.println("Player is " + pig.getActualPlayer());
        while (!done) {
            System.out.print("> ");
            String cmd = s.nextLine();
            switch (cmd) {
                case "r":
                    int result = pig.roll();
                    System.out.println(result);
                    if (result == 1) {
                        dump(pig);
                        pig.next();
                        System.out.println("Player is " + pig.getActualPlayer());
                    }
                    break;
                case "n":
                    pig.setTotal();
                    dump(pig);
                    if (pig.gameOver()) {
                        done = true;
                    } else {
                        pig.next();
                        System.out.println("Player is " + pig.getActualPlayer());
                    }
                    break;
                case "q":
                    done = true;
                    break;
                default:
                    System.out.println("?");
            }
        }
        if( pig.gameOver() ){
            System.out.println("Game over! Winner is " + pig.getActualPlayer());
        }else{
            dump(pig);
            System.out.println("Game aborted");
        }
    }

    private static void dump(Pig pig) {
        for (Player p : pig.getPlayers()) {
            System.out.print(p + " ");
        }
        System.out.println();
    }
}
```

d)

```
import java.util.Random;
public abstract class AbstractDice {
    private final int faces;
    private final Random random = new Random();
    private int result;

    public AbstractDice(int faces) {
        this.faces = faces;
    }

    public int roll() {
        result = random.nextInt(faces) + 1;
        return result;
    }

    public int getLastResult() {
        if (result > 0) {
            return result;
        } else {
            throw new IllegalStateException("Must roll dice first");
        }
    }
}


public class PigDice extends AbstractDice {
    private int total;

    public PigDice(int faces) {
        super(faces);
    }

    @Override
    public int roll() {
        int result = super.roll();
        if (result > 1) {
            total += result;
        } else {
            total = 0;
        }
        return result;
    }

    public int getTotal() {
        return total;
    }

    public void clear() {
        total = 0;
    }
}
```

Uppgift 6

```
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
public class MainFrame extends JFrame implements ActionListener {
    private final JTextField text = new JTextField("In");
    private final JLabel label = new JLabel("Out");
    public MainFrame() {
        this.setLayout(new GridLayout(2, 2));
        this.add(text);
        this.add(label);
        text.addActionListener(this);
        this.setSize(150, 150);
    }

    public static void main(String[] args) {
        new MainFrame().setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JTextField t = (JTextField) e.getSource();
        label.setText(t.getText());
    }
}
```

```
public class Player {
    private final String name;
    private int total;

    public Player(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public int getTotal() {
        return total;
    }

    public void addPoints(int points) {
        total += points;
    }

    @Override
    public String toString() {
        return "{" + name + " : " + total + "}";
    }
}//Player
```

```
import java.util.ArrayList;
import java.util.List;
public class PigOptions {

    private PigOptions() {}

    private static final int POINTS_TO_WIN = 20;  //During development
    // User options
    private static final List<Player> players = new ArrayList<>();
    private static int pointsToWin = POINTS_TO_WIN;

    public static List<Player> getPlayers() {
        // Default players
        players.add(new Player("Anna"));
        players.add(new Player("Urban"));
        players.add(new Player("Fia"));
        return players;
    }

    public static int getPointsToWin() {
        return pointsToWin;
    }

    public static void setPointsToWin(int pts) {
        pointsToWin = pts;
    }
}
```

```
public class Point {
    private int x;
    private int y;
    public Point (int x, int y) {
        this.x = x;
        this.y = y;
    }//constructor
    public int getX () {
        return x;
    }//getX

    public int getY () {
        return y;
    }//getY

    public Point translate(int dx, int dy) {
        return new Point(x + dx, y +dy);
    }//translate

    public double distance(Point p) {
        return Math.sqrt(Math.pow(x-p.x, 2) + Math.pow(y-p.y, 2));
    }//distance

    public boolean equals(Point p) {
        return (x == p.x && y == p.y);
    }//equals

    public String toString() {
        return "(" + x + ", " + y + ")";
    }//toString
}//Point
```