Lösningsförslag till tentamen 140828

Uppgift 1

a)

```
public class MyMax {
    public static void main(String[] args) {                          //5  ändra String till String[]
        System.out.println("Maximum is: " + maximum(5, 7));           //2 ändra "," till "+"
        System.out.println("Maximum is: " + maximum(7, 5));           //2 ändra "," till "+"
        System.out.println("Maximum is: " + maximum(3, 4, 9));        //2 ändra "," till "+"
        System.out.println("Maximum is: " + maximum(3, 9, 4));        //2 ändra "," till "+"
        System.out.println("Maximum is: " + maximum(9, 4, 3));        //2 ändra "," till "+"
    }//constructor

    // Returns the maximun value of the parameters x, y and z
    public static int maximum (int x, int y, int z) {                 //3  måste vara en statisk metod, lägg till static
        int max = x;
        if (y >  max)                                                 //6 ta bort ";"
            max = y;
        if (z > max)
            max = z;
        return max;
    }//maximum

    // Returns the maximun value of the parameters x and y
    public static int maximum (int x, int y) {                        //1 varje parameter måste anges med en typ
        if (x > y)
            return x;
        return y;                                                     //4  y måste returneras om inte x är störst
    }//maximum
}//MyMax
```

b)

När metoden grow anropas uppdateras inte instansvariabeln area!

Felet kan korrigeras antingen genom att förändra metoden grow:

```
public void grow() {
    sideLength = 2 * sideLength;
    area = sideLength*sideLength;
}//grow
```

Man kan dock diskutera om instansvariabeln area överhuvudtaget skall finnas i klassen. En (bättre) alternativ lösning är därför

```
public class Square {
    private int sideLength;

    public Square(int initiallength) {
        sideLength = initiallength;
    }//constructor

    public int getArea() {
        return sideLength * sideLength;
    }//getArea

    public void grow() {
        sideLength = 2 * sideLength;
    }//grow
}//Square
```

c)

Felet är att likhetsoperatorn == inte kan användas, eftersom likhetsoperatorn == ger **true** om och endast om operanderna avser samma objekt! För att få det önskade resultatet skall objekten jämföras med equals-metoden.

```
public static boolean proceed() {
      Scanner scannerObject = new Scanner(System.in);
      while (true) {
          System.out.println("Answer yes to continue, and no to stop.");
          String answer = scannerObject.next();
          if (answer.equals("yes") || answer.equals("y"))
                return true;
          if (answer.equals("no") || answer.equals("n"))
               return false;
          else
               System.out.println("Your answer should be yes or no!");
      }
}//proceed
```

Uppgift 2

Med användning av dialogrutor:

```
import javax.swing.*;
import java.util.*;
public class Heron {
    public static void main (String[] arg) {
        while (true) {
            String indata = JOptionPane.showInputDialog("Ange längden på triangelns tre sidorna:");
            if (indata == null)
                break;
            Scanner sc = new Scanner(indata);
            double a = sc.nextDouble();
            double b = sc.nextDouble();
            double c = sc.nextDouble();
            if (a >= (b+c) || b >= (a+c) || c >= (a+b) ||) {
                JOptionPane.showMessageDialog(null, "DETTA ÄR INGEN TRIANGEL!");
            }
            else {
                double yta = heron(a, b, c);
                JOptionPane.showMessageDialog(null, String.format("Ytan av triangeln är %.3f", yta));
            }
        }
    }
    public static double heron(double a, double b, double c) {
        double p = (a+b+c) / 2.0;
        return Math.sqrt(p*(p-a)*(p-b)*(p-c));
    }//heron
}//Heron
```

Med användning av System.in och System.out:

```
import java.util.*;
public class Heron {
    public static void main (String[] arg) {
        while (true) {
            System.out.print("Ange längden på triangelns tre sidorna: ");
            Scanner sc = new Scanner(System.in);
            if (!sc.hasNext())
                    break;
            double a = sc.nextDouble();
            double b = sc.nextDouble();
            double c = sc.nextDouble();
            if (a >= (b+c) || b >= (a+c) || c >= (a+b)) {
                System.out.println("DETTA ÄR INGEN TRIANGEL!");
            }
            else {
                double yta = heron(a, b, c);
                System.out.println(String.format("Ytan av triangeln är %.3f", yta));
            }
        }
    }//main
    public static double heron(double a, double b, double c) {
        double p = (a+b+c) / 2.0;
        return Math.sqrt(p*(p-a)*(p-b)*(p-c));
    }//heron
}//Heron
```

Uppgift 3

```
public static int[] append(int[] vektA, int[] vektB)  {
    if (vektA == null && vektB == null)
        throw new IllegalArgumentException();
    if (vektA == null) {
        int[] res = new int[vektB.length];
        for (int i = 0; i < vektB.length; i = i + 1) {
            res[i] = vektB[i];
        }
        return res;
    } else if (vektB == null) {
        int[] res = new int[vektA.length];
        for (int i = 0; i < vektA.length; i = i + 1) {
            res[i] = vektA[i];
        }
            return res;
    } else {
        int[] res = new int[vektA.length + vektB.length];
        int index = 0;
        for (int i = 0; i < vektA.length; i = i + 1) {
            res[index] = vektA[i];
            index = index + 1;
        }
        for (int i = 0; i < vektB.length; i = i + 1) {
            res[index] = vektB[i];
            index = index + 1;
        }
        return res;
    }
}//append
```

Alternativ lösning:

```
public static int[] append(int[] vektA, int[] vektB)  {
    if (vektA == null && vektB == null)
        throw new IllegalArgumentException();
    if (vektA == null) {
        vektA = new int[0];
    } else if (vektB == null) {
        vektB = new int[0];
    }
    int[] res = new int[vektA.length + vektB.length];
    int index = 0;
    for (int i = 0; i < vektA.length; i = i + 1) {
        res[index] = vektA[i];
        index = index + 1;
    }
    for (int i = 0; i < vektB.length; i = i + 1) {
        res[index] = vektB[i];
        index = index + 1;
    }
    return res;
}//append
```

Uppgift 4

```
public static int[][][] sepiaFilter(int[][][] sample) {
    int[][][] newSample = new int[sample.length][sample[0].length][3];
    for (int x = 0; x < newSample.length; x = x + 1) {
        for (int y = 0; y < newSample[x].length; y = y + 1) {
            newSample[x][y][0] = (int) luminans(sample[x][y]);
            newSample[x][y][1] = (int) (0.6 * luminans(sample[x][y]));
            newSample[x][y][2] = (int) (0.4 * luminans(sample[x][y]));
        }
    }
    return newSample;
}//sepiaFilter

public static double luminans(int[] c) {
    return 0.299*c[0] + 0.587*c[1] + 0.114*c[2];
}//luminans
```

Uppgift 5

a)

```
import java.util.Random;
public class MexicoDice {
    private final Random random = new Random();
    public int roll() {
        int r1 = random.nextInt(6) + 1;
        int r2 = random.nextInt(6) + 1;
        int result;
        if (r1 >= r2) {
            result = 10 * r1 + r2;
        } else {
            result = 10 * r2 + r1;
        }
        return result;
    }//roll
}//MexicoDice
```

b)

```
    public Mexico(List<String> playerNames) {
        MexicoDice dice = new MexicoDice();
        for (String name : playerNames) {
            players.add(new Player(name, MexicoOptions.STARTING_FEE , dice));
        }
    }//constructor
```

c)

```
    public boolean finished() {
        int count = 0;
        for (Player p : players) {
            if (p.hasMoney()) {
                count++;
            }
        }
        return count == 1;
    }//finished
```

d)

```
    public void potToWinner() {
        for (Player p : players) {
            if (p.hasMoney()) {
                p.add(pot);
                pot = 0;
                break;
            }
        }
    }//potToWinner
```

e)
```
public Player lowestGrade() {
    final int MAX = 100000;
    List<Integer> scores = new ArrayList<>();
    for (Player p : players) {
        if (p.getScore() == 0 || p.getScore() == MEXICO) {
            scores.add(MAX);
        } else if (p.getScore() / 10 == p.getScore() % 10) {
            int i = p.getScore();
            scores.add(100 * i + 10 * i + i);
        } else {
            scores.add(p.getScore());
        }
    }
    int minIndex = 0;
    for (int i = 0; i < scores.size(); i++) {
        if (scores.get(i) < scores.get(minIndex)) {
            minIndex = i;
        }
    }
    return players.get(minIndex);
}//lowestGrade
```
Alternativ lösning:
```
public Player lowestGrade() {
    final int MAX = 100000;
    int minValue = MAX;
    Player looser = null;
    for (Player p : players) {
        int value;
        if (p.getScore() == 0 || p.getScore() == MEXICO) {
            value = MAX;
        } else if (p.getScore() / 10 == p.getScore() % 10) {
            value  = 10 * p.getScore();
        } else {
            value = p.getScore();
        }

        if (value  < minValue) {
            minValue = value;
            looser = p;
        }
    }
    return looser;
}//lowestGrade
```
f)
```
        Player p = mexico.lowestGrade();
        System.out.println(p.getName() + " must put in pot");
        p.withdraw(MexicoOptions.PAYOFF) ;
        mexico.addToPot(MexicoOptions.PAYOFF );
```