Lösningsförslag till tentamen 150418

Uppgift 1

a)

i) Utskriften blir:

[8, 3, 9, 4, 5]

ii) Metoden mystery flyttar om elementen i fältet arr på så sätt att alla elementen, förutom det första, flyttas ett position till vänster. Det första elementet hamnar sist.

iii) Exempelvis

```
public static void shiftLeft(int[] arr) {
    for (int i = 0; i < arr.length - 1; i = i + 1) {
        int temp = arr[i];
        arr[i] = arr[i + 1];
        arr[i + 1] = temp;
    }
}//shiftLeft
```

eller

```
public static void shiftLeft(int[] arr) {
    int temp = arr[0];
    for (int i = 0; i < arr.length - 1; i = i + 1) {
        arr[i] = arr[i + 1];
    }
    arr[arr.length-1] = temp;
}//shiftLeft
```

b) Utskriften blir:

1, 2, 6
4, 5, 6

c) Precis som kompilatorn påpekar i sitt felmeddelande saknas det en **return**-sats. Eftersom parametern betyg är av typen int kan metoden anropas med alla existerande värden av typen **int**, inte bara värdena 0 till 4 som Stina använder. Detta betyder att alla värden förutom värdena 0 till 4 "passerar förbi" samtliga **if**-satser och följaktligen stöter dessa värden inte på någon **return**-sats.

Det finns flera sätt att korrigera metoden på. En bra princip, för alla icke-**void**-metoder, är att ha endast en **return**-sats som ligger allra sist i metoden (alla exekveringsvägar i metoden leder ju i så fall dit). En lämplig modifiering av metoden är därför:

```
public static String omvandla(int betyg) {
    String utdata = "";
    if (betyg == 0)
        utdata = "Urdålig!";
    else if (betyg == 1)
        utdata = "Dålig!";
    else if (betyg == 2)
        utdata = "Godkänd!";
    else if (betyg == 3)
        utdata = "Bra!";
    else if (betyg == 4)
        utdata = "Jättebra!";
    return utdata;
}//omvandla
```

Uppgift 2

```
import javax.swing.*;
import java.util.*;
public class Growth {
    public static void main (String[] arg) {
        while (true) {
            String indata = JOptionPane.showInputDialog("Ange kapital, ränta och antal år:");
            if (indata == null)
               break;
            Scanner sc = new Scanner(indata);
            double k = sc.nextDouble();
            double p = sc.nextDouble();
            double t = sc.nextDouble();
            if (k < 0 || p < 0 || t < 0)
               JOptionPane.showMessageDialog(null, "Ogiltig indata!");
            else {
               JOptionPane.showMessageDialog(null, String.format("%.2f kronor kommer med en ränta på ", k)
                                                 + String.format("%.2f procent\n" , p)
                                                 + String.format("att på %.2f år växa med %.2f kronor", t, interest(k,p,t)));
            }
        }//while
    } //main

     private static double interest(double capital, double rate, double year) {
          return  capital * Math.pow(1 + rate/100, year) - capital;
     }
} //Growth
```

Uppgift 3

```
    public static boolean containsAsSequence(int[] source, int[] target) {
    if (source == null || target == null)
        throw new IllegalArgumentException();
    boolean found = false;
    for (int i = 0; i < source.length - target.length; i = i + 1) {
        boolean okey = true;
        for (int j = 0; j < target.length; j = j + 1) {
            if (source[i + j] != target[j])
               okey = false;
        }
        if (okey)
            found = true;
    }
    return found;
}//containsAsSequence
```

Uppgift 4

```
public static int[][][] montage(int[][][] foreground, int[][][] background) {
    int[][][] newSample = new int[foreground.length][foreground[0].length][3];
    for (int row = 0; row < newSample.length; row = row + 1) {
        for (int col = 0; col < newSample[x].length; col = col + 1) {
            if (isGreen(row, col, foreground)) {
               for (int i = 0; i < 3; i = i + 1) {
                    newSample[row][col][i] = background[row][col][i];
               }
            }
            else {
               for (int i = 0; i < 3; i = I + 1) {
                    newSample[x][y][i] = foreground[x][y][i];
               }
            }
        }
    }
    return newSample;
}//montage

private static boolean isGreen(int i, int j, int[][][] image) {
     return image[i][j][0] == 0 && image[i][j][1] == 255 && image[i][j][2] == 0 ;
}//isGreen
```

Uppgift 5

```
public String generate(String s, int n) {
    Random r = new Random();
    String rStr = "";
    if (s.length() > 0) {
        for (int i = 0; i < n; i++) {
            int j = r.nextInt(s.length());
            char ch = s.charAt(j);
            rStr = rStr + ch;
        }
    }
    return rStr;
}
```

Uppgift 6

```
public class PLang {
    public String toP(String swe) {
        String p = "";
        for (char ch : swe.toCharArray()) {
            if (isVowel(ch)) {
               p = p + ch + "p" + ch;
            } else {
               p = p + ch;
            }
        }
        return p;
    }

    private boolean isVowel(char ch) {
        return "aouåeiyäö".indexOf(ch) >= 0;
    }
}
```

Uppgift 7

Exempel på olika tänkbara klasser och attribut (finns fler)

```
public class Player {
   private final String name;
   private final Piece piece;
   public Player(String name, Piece piece) {
      this.name = name;
      this.piece = piece;
   }
}

public class Piece {
   private final Color[] wedges = new Color[6];
   public void put(Color wedge) {
      // If not there and not full, ... add it
   }
}

public class Card {
   private final Board.Topic topic;
   private final String text;
   public Card(Board.Topic topic, String text) {
      this.topic = topic;
      this.text = text;
   }
}

public class Board {
   public enum Topic {
       Geography, Entertainment, History,  ArtsAndLiterature, ScienceAndNature, SportsAndLeisure
   }
   private final Space[] spaces;
   public Board(Space[] spaces) {
      this.spaces = spaces;
   }
}

public class Space {
   private final Topic topic;
   public Space(Topic topic) {
      this.topic = topic;
   }
}
```

Så här kan man konstruera modellen

```
private TrivialPursuit buildTP() {
      Player[] players = { new Player("olle", new Piece()), new Player("fia", new Piece()) /* ...etc... */ };
      Space[] spaces = { new Space(Board.Topic.History) /* ...etc ..*/ };
      Board board = new Board(spaces);
      Card[] cards = { new Card(Board.Topic.History, "Vad hade Gustav ...") /* ... etc ..*/ };
      return new TrivialPursuit(players, board, cards);
   }
```

Uppgift 8

```
private void advice(int[] a, int[] b) {
    boolean success = false;
    for (int i = 0; i < a.length; i++) {
        if (a[i] < getMean(a) && a[i] > getMean(b)) {
            System.out.println("Move " + a[i] + " from A to B");
            success = true;
        }
    }
    for (int i = 0; i < b.length; i++) {
        if (b[i] < getMean(b) && b[i] > getMean(a)) {
            System.out.println("Move " + b[i] + " from B to A");
            success = true;
        }
    }
    if (!success) {
        System.out.println("Omöjligt");
    }
}//advice

private double getMean(int[] arr) {
    double sum = 0;
    for (int i : arr) {
        sum = sum + i;
    }
    return sum / arr.length;
}//getMean
```