Lösningsförslag, kontrollskrivning 3 PTDC

2011–12–12

1.
```
public class NormalRandom {
    private static Random rand = new Random();
    private double z1;
    private boolean haveZ1;

    public NormalRandom() {
        haveZ1 = false;
    }

    public double nextNormal() {
        if (haveZ1) {
            haveZ1 = false;
            return z1;
        }
        double u0 = 1 - rand.nextDouble();
        double u1 = 1 - rand.nextDouble();
        double r = Math.sqrt(-2 * Math.log(u0));
        double theta = 2 * Math.PI * u1;
        double z0 = r * Math.cos(theta);
        z1 = r * Math.sin(theta);
        haveZ1 = true;
        return z0;
    }
}
```

2.
```
public class BlogPost {
    private String title;
    private String text;
    private String date;
    private ArrayList<String> tags;

    public BlogPost(String title, String text, String date) {
        this.title = title;
        this.text = text;
        this.date = date;
        tags = new ArrayList<String>();
    }

    public void addTag(String tag) {
        if (!tags.contains(tag)) {
            tags.add(tag);
        }
    }
```

```
    public boolean allTagsMatch(ArrayList<String> words) {
        for (int i = 0; i < words.size(); i++) {
            if (!tags.contains(words.get(i))) {
                return false;
            }
        }
        return true;
    }

    public String toString() {
        return title + " " + date + "\n" + text;
    }
}
```

3.
```
    public ArrayList<BlogPost> getMatchingPosts(ArrayList<String> words) {
        ArrayList<BlogPost> result = new ArrayList<BlogPost>();
        for (int i = 0; i < posts.size(); i++) {
            if (posts.get(i).allTagsMatch(words)) {
                result.add(posts.get(i));
            }
        }
        return result;
    }
```

4.
```
public class Polynom {
    private ArrayList<Term> terms;

    public Polynom() {
        terms = new ArrayList<Term>();
    }

    public void addTerm(double coeff, int degree) {
        if (coeff == 0) {
            return;
        }
        int pos = findGreaterEqual(degree);
        if (pos < terms.size() && terms.get(pos).getDegree() == degree) {
            if (terms.get(pos).getCoeff() + coeff != 0) {
                terms.get(pos).addToCoeff(coeff);
            } else {
                terms.remove(pos);
            }
        } else {
            terms.add(pos, new Term(coeff, degree));
        }
    }

    private int findGreaterEqual(int degree) {
        for (int i = 0; i < terms.size(); i++) {
            if (terms.get(i).getDegree() >= degree) {
                return i;
            }
        }
        return terms.size();
    }
```

```
    public double getValue(double x) {
        double sum = 0;
        for (int i = 0; i < terms.size(); i++) {
            Term t = terms.get(i);
            sum += t.getCoeff() * Math.pow(x, t.getDegree());
        }
        return sum;
    }

    public Polynom differentiate() {
        Polynom res = new Polynom();
        for (int i = 0; i < terms.size(); i++) {
            Term t = terms.get(i);
            if (t.getDegree() > 0) {
                res.terms.add(new Term(t.getDegree() * t.getCoeff(),
                    t.getDegree() - 1));
            }
        }
        return res;
    }

    public void print() {
        for (int i = 0; i < terms.size(); i++ ) {
            Term t = terms.get(i);
            System.out.println(t.getCoeff() + " " + t.getDegree());
        }
    }
}
```