Hausaufgabe 6.1: Hashfunktionen

```
package hashfunctions;

import java.text.NumberFormat;

// Allgemeine Hashfunktion-Klasse
public abstract class Hashfunction {

    // Implementierung in a) und b)
    public abstract int hash(int key);

    // nur für Ausgabe der Hashwerte
    public static String usage() {
        StringBuffer usage = new StringBuffer();
        usage.append("Prints the value of the given Integers ");
        usage.append("as assigned by the specified Hashfunction.\n");
        usage.append("first argument: classname extending ");
        usage.append(Hashfunction.class.toString());
        usage.append("\n");
        usage.append("second and following arguments: Integer (key)\n");
        return usage.toString();
    }

    public static void main(String[] args) {
        try {
            // neue Hashfunktion wird erzeugt (aus erstem Argument)
            Hashfunction hashfunction = (Hashfunction)
                        Class.forName(args[0]).newInstance();

            // Hashwerte sollen immer 3 Ziffern haben
            NumberFormat hashformat = NumberFormat.getIntegerInstance();
            hashformat.setMinimumIntegerDigits(3);

            // Kopieren der Schlüssel aus restlichen Aufrufparametern
            int[] keys = new int[args.length-1];
            for(int i = 0; i < keys.length; i++)
                keys[i] = Integer.parseInt(args[i+1]);
            StringBuffer result = new StringBuffer();
            // Ausgabe jedes Schlüssels mit dem zugehörigen Hashbucket
            for(int i = 0; i < keys.length; i++)
                result.append("Key: ");
                result.append(keys[i]);
                result.append(" - Hash: ");
                result.append(hashformat.format
                                    (hashfunction.hash(keys[i])));
                result.append('\n');
            }
            System.out.print(result);
        }
        catch(Exception e) {
            System.err.println(e);
            System.err.println(usage());
        }
    }
}
```

a) Middle Square

```
public class MiddleSquare extends Hashfunction {

    // Gesucht werden hier die mittleren beiden Stellen (3 und 4) des
    // quadrierten Schlüssels. Gesucht werden nur zwei Stellen, da sonst
    // Hashwerte > 255 möglich wären; allerdings sind hier nun nur
    // Hashwerte < 100 (da zweistellig) möglich.
    public int hash(int key) {
        int keySquared = key*key; // quadrierter Wert
        int first = (keySquared / 10000) % 10; // erste Stelle
        int second = (keySquared / 1000) % 10; // zweite Stelle
        return first*10 + second; // erste Stelle ist Zehnerstelle
    }
}
```

b) Division

```
public class Division extends Hashfunction {
    private int m = 251;

    public Division() {}

    public Division(int m) {
        this.m = m;
    }

    public int hash(int key) {
        return key % m;
    }
}
```