HW 4 – Hashing

Rehashing requires recomputing the hash function for all items in the hash table. Since computing the hash function is expensive, suppose objects provide a hash member function of their own, and each object stores the result in an additional data member the first time the hash function is computed for it. Show how such a scheme would apply for the Employee class below, and explain under what circumstances the remembered hash value remains valid in each Employee.

```
public class Employee {

    private String name;
    private double salary;
    private int seniority;


    public Employee(String name, double salary, int seniority) {
        this.name = name;
        this.salary = salary;
        this.seniority = seniority;
    }

    public boolean equals( Object rhs )

         { return rhs instanceof Employee && name.equals( ((Employee)rhs).name ); }

    public int hashCode( )
           { return name.hashCode( ); }



}
```

Bonus (30 points): implement this idea by wrapping the universal hashing function logic into a function object called a HashFunction, which can be used to generate this new, stored hash of the Employee. Give the Employee another instance variable which holds one of these HashFunction objects, and use that object to do the work of rehashing for you…