```java
package datamanagment;

import java.util.HashMap;

public class TaxCalculator {
    private float incomeTiers[];
    private float fixedTaxTiers[];
    private float percentageModifierTiers[];
    private int maxTier;

    private TaxCalculator(float incomeTiers[], float fixedTaxTiers[], float percentageModifierTiers[]) {
        this.incomeTiers = incomeTiers.clone();
        this.fixedTaxTiers = fixedTaxTiers.clone();
        this.percentageModifierTiers = percentageModifierTiers.clone();
        maxTier = this.incomeTiers.length - 1;
    }

    private static HashMap<String, TaxCalculator> taxCategoryMap = new HashMap<String, TaxCalculator>() {
        private static final long serialVersionUID = 1L;

        {
            put("Married Filing Jointly",
                    new TaxCalculator(new float[] { 0, 36080, 90000, 143350, 254240 },
                            new float[] { 0, 1930.28f, 5731.64f, 9492.82f, 18197.69f },
                            new float[] { 0.0535f, 0.0705f, 0.0705f, 0.0785f, 0.0985f }));

            put("Married Filing Seperately",
                    new TaxCalculator(new float[] { 0, 18040, 71680, 90000, 127120 },
                            new float[] { 0, 965.14f, 4746.76f, 6184.88f, 9098.80f },
                            new float[] { 0.0535f, 0.0705f, 0.0785f, 0.0785f, 0.0985f }));

            put("Single",
                    new TaxCalculator(new float[] { 0, 24680, 81080, 90000, 152540 },
                            new float[] { 0, 1320.38f, 5296.58f, 5996.80f, 10906.19f },
                            new float[] { 0.0535f, 0.0705f, 0.0785f, 0.0785f, 0.0985f }));

            put("Head of Household",
                    new TaxCalculator(new float[] { 0, 30390, 90000, 122110, 203390 },
                            new float[] { 0, 1625.87f, 5828.38f, 8092.13f, 14472.61f },
                            new float[] { 0.0535f, 0.0705f, 0.0705f, 0.0785f, 0.0985f }));
        }
    };

    public float calculateTax(float income) {
        int tier = maxTier;
        while (income < incomeTiers[tier]) {
            tier--;
        }
        return fixedTaxTiers[tier] + percentageModifierTiers[tier] * (income - incomeTiers[tier]);
    }

    public static TaxCalculator getTaxCalculator(String status) {
```

```
53          return taxCategoryMap.get(status);
54      }
55
56      public float calculateTaxIncrease(float income, float tax, float receiptsValue) {
57          float increaseTiers[] = { 0, 0.2f, 0.4f, 0.6f };
58          float percentageTiers[] = { 0.08f, 0.04f, -0.15f, -0.3f };
59          int tier = 3;
60          while (receiptsValue < increaseTiers[tier] * income) {
61              tier--;
62          }
63          return tax * percentageTiers[tier];
64      }
65  }
66
```