Command Prompt

```
               5 File(s)        387,331 bytes
               5 Dir(s)  118,842,134,528 bytes free

C:\Users\MEDITEX\eclipse-workspace\refactor>javac -cp ./junit-4.13.2.jar;. WriteAUnitTest.java
WriteAUnitTest.java:16: error: illegal character: '\u2013'
                return ( compoundInterest ? principal );
                                          ^
WriteAUnitTest.java:16: error: not a statement
                return ( compoundInterest ? principal );
                                            ^
WriteAUnitTest.java:16: error: ';' expected
                return ( compoundInterest ? principal );
                                                      ^
WriteAUnitTest.java:29: error: class, interface, enum, or record expected
 float calculateFee(Account accounts[]) {
 ^
WriteAUnitTest.java:31: error: class, interface, enum, or record expected
         Account account;
         ^
WriteAUnitTest.java:32: error: class, interface, enum, or record expected
         for (int i = 0; i < accounts.length; i++) {
         ^
WriteAUnitTest.java:32: error: class, interface, enum, or record expected
         for (int i = 0; i < accounts.length; i++) {
                         ^
WriteAUnitTest.java:32: error: class, interface, enum, or record expected
         for (int i = 0; i < accounts.length; i++) {
                                              ^
WriteAUnitTest.java:34: error: class, interface, enum, or record expected
                 if ( account.isPremium( ) ) {
                 ^
WriteAUnitTest.java:36: error: class, interface, enum, or record expected
                        }
                        ^
WriteAUnitTest.java:39: error: class, interface, enum, or record expected
 }
 ^
WriteAUnitTest.java:41: error: class, interface, enum, or record expected
static final double BROKER_FEE_PERCENT = 0.0125;
            ^
12 errors

C:\Users\MEDITEX\eclipse-workspace\refactor>
```

```
class Account {
    float principal;
    float rate;
    int daysActive;
    int accountType;

    public static final int STANDARD = 0;
    public static final int BUDGET = 1;
    public static final int PREMIUM = 2;
    public static final int PREMIUM_PLUS = 3;

    @Test
    float interestEarned() {
        float years = daysActive / (float) 365.25;
        float compoundInterest = principal * (float) Math.exp( rate * years );
        return ( compoundInterest – principal );
        }
    @Test
    public boolean isPremium() {
        if (accountType == Account.PREMIUM || accountType == Account.PREMIUM_PLUS) {
             return true;
        }else {
             return false;
        }
    }
}

@Test
 float calculateFee(Account accounts[]) {
     float totalFee = 0;
     Account account;
     for (int i = 0; i < accounts.length; i++) {
         account = accounts[i];
         if ( account.isPremium( ) ) {
             totalFee += BROKER_FEE_PERCENT * account.interestEarned( );
             }
         }
     return totalFee;
 }

static final double BROKER_FEE_PERCENT = 0.0125;
```