BANK ACCOUNT

```
class BankAccount {

  private static final double HIGHER_INTEREST_PERCENT = 0.05;
  private static final double SERVICE_CHARGE = 10.00;
  private static final double LOWER_THRESHOLD = 1000.00;
  private static final double HIGHER_THRESHOLD = 10000.00;
  private static final double LOWER_INTEREST_PERCENT = 0.04;

  double accountBalance = 0.00;

  public BankAccount (double accountBalance) {
    this.accountBalance = accountBalance;
  }

  public boolean incursServiceCharge() {
    boolean balanceUnderOneThousand = (this.accountBalance <
LOWER_THRESHOLD);
    return balanceUnderOneThousand;
  }

  public double serviceChargeAmt() {
    if (incursServiceCharge()) {
    return SERVICE_CHARGE;
    } else {
      return 0d;
    }
  }

  public double interestEarned() {
    if ((this.accountBalance >= LOWER_THRESHOLD) && (this.accountBalance <
HIGHER_THRESHOLD)) {
      return interestOneThousandTenThousand();
    } else if (this.accountBalance >= HIGHER_THRESHOLD) {
      return interestOverTenThousand();
    } else if (this.accountBalance < LOWER_THRESHOLD) {
      return 0d;
    }
      return 0d;
    }

    private boolean isInterestOverTenThousand() {
      boolean isLoaded = (this.accountBalance > HIGHER_THRESHOLD);
      return isLoaded;

    }
    private double interestOneThousandTenThousand() {
      double interestAccruedBtwOnekTenk = this.accountBalance *
LOWER_INTEREST_PERCENT;
      if (isInterestOverTenThousand()) {
        return HIGHER_THRESHOLD * LOWER_INTEREST_PERCENT;
      } else {
        return interestAccruedBtwOnekTenk;
      }
  }
    private double interestOverTenThousand() {
    double balanceAfterFirstTenk = (this.accountBalance - HIGHER_THRESHOLD);
    double interestAccruedOverTenk = (balanceAfterFirstTenk *
HIGHER_INTEREST_PERCENT);
    double balanceOverTenk = (interestOneThousandTenThousand() +
interestAccruedOverTenk);
    return balanceOverTenk;
    }
  }
```

Give 2 suggestions that would make this code more expressive.

COFFEE DRINKER

```
class FancyCoffeeCup {

  int cupCapacity;
  int currCoffee;



  public FancyCoffeeCup(int cupCapacity) {
  this.cupCapacity = cupCapacity;
  currCoffee = cupCapacity;
  }



  public int sip(int slurp) {
  if (slurp > currCoffee) {
  slurp = currCoffee % slurp;
  currCoffee = Math.max(currCoffee - slurp, 0);
  return slurp;
  } else {
  currCoffee -= slurp;
  return slurp;
  }


  }

  public String toString(){
  return "" + currCoffee + "!";
  }
}


class Person {
String name;
int defaultSipAmt;
int mlsInBelly = 0;
private FancyCoffeeCup fancyCoffeeCup;


public Person(String name, int defaultSipAmt) {
this.name = name;
this.defaultSipAmt = defaultSipAmt;
}


public void sip(FancyCoffeeCup fancyCoffeeCup) {
this.fancyCoffeeCup = fancyCoffeeCup;
mlsInBelly += fancyCoffeeCup.sip(defaultSipAmt);
}
public void gulp(FancyCoffeeCup fancyCoffeeCup) {
this.fancyCoffeeCup = fancyCoffeeCup;
mlsInBelly += fancyCoffeeCup.sip((defaultSipAmt * 2));
}


public int amountCoffeeDrunk() {
return mlsInBelly;
}
```

Give 2 suggestions that would make this code more expressive.

PASSWORD VALIDATOR 1

```
class PasswordValidator {

  private int minPasswordLen;
  private int numSpecialChars;
  private int numDigits;
  private int numUppercase;


  public PasswordValidator(int minPasswordLen, int numSpecialChars, int
numDigits, int numUppercase) {
    this.minPasswordLen = minPasswordLen;
    this.numSpecialChars = numSpecialChars;
    this.numDigits = numDigits;
    this.numUppercase = numUppercase;
  }


  public boolean validated(String passwordCandidate) {
    int passwordLength = passwordCandidate.length();

    if (passwordCandidate.contains(" ")) {
      return false;
    }

    if (passwordLength >= minPasswordLen &&
numSpecialChars(passwordCandidate) >= numSpecialChars &&
        numDigits(passwordCandidate) >= numDigits &&
numUppercase(passwordCandidate) >= numUppercase) {
          return true;
        } else {
          return false;
        }
  }

}
```

1. Give 2 suggestions that would make this code more expressive.
2. Do you find this version or version 2 easier to understand? Why?

PASSWORD VALIDATOR 2

```
class PasswordValidator {

  // you'll need some instance variables....
  private int minNumChars ;
  private int minNumSpecialChars ;
  private int minNumDigits ;
  private int minNumUpperChars ;

  private boolean check;


  public PasswordValidator(int minPasswordLen, int numSpecialChars, int
numDigits, int numUppercase) {
  this.minNumChars = minPasswordLen ;
  this.minNumSpecialChars = numSpecialChars ;
  this.minNumDigits = numDigits ;
  this.minNumUpperChars = numUppercase ;

  }


  public boolean validated(String passwordCandidate) {
    if ( (numSpecialChars(passwordCandidate) >= minNumSpecialChars)
      && (numDigits(passwordCandidate) >= minNumDigits)
      && (numUppercase(passwordCandidate) >= minNumUpperChars)
      && (passLength(passwordCandidate) >= minNumChars)
      && (spaceChecker(passwordCandidate) == false) ){

     check = true ;

    } else {
      check = false ;
    }

    return check ;

  }

  // Returns the number of special (not letters of the alphabet or digits)
  // characters in a given string.
  //

  private boolean spaceChecker(String passwordCandidate){
    boolean spaceCheck = false ;
  if (passwordCandidate.contains(" ") ) {
    spaceCheck = true ;
  }
  return spaceCheck ;

  }


  private int passLength(String passwordCandidate){
   int charCount = passwordCandidate.length() ;

    return charCount ;
  }


}
```

1. Give 2 suggestions that would make this code more expressive.
2. Do you find this version or version 1 easier to understand? Why?