AP Computer Science
Fall Final

Name _____
Period _____

Part I – Multiple Choice

Answers

1. D
2. B
3. D
4. C
5. B
6. D
7. B
8. E
9. D
10. C
11. A
12. E
13. C
14. B
15. D (omitted)
16. D
17. B

Free Response

Name _____

1. (15 points) The following class models a utility customer much like a customer of the City of Healdsburg. You should complete the following methods. The description of each method is below

```
public class UtilityCustomer
{
      private String myName;  // Name of customer in Last, First format
                              // example: Efram, Mike

      private String myPhone; // Phone number in ###-###-#### format
                              // example: 707-433-5777

      private double myElecRate;    // rate per kilowatt-hour (kwh) of electricity
      private double myWaterRate;   // rate per unit of water

      private boolean isResdential; // true if residential customer
                                    // false if commercial customer

      // Default Constructor: This constructor initializes a utility customer to
      // a default setting. The following should be set as the default customer
      // myName should be set to an empty String, ""
      // myPhone should be set to "000-000-0000"
      // isResidential should be set to true
      // myElecRate should be set to $0.12
      // myWaterRate should be set to $2.35
      public UtilityCustomer()
      {
            myName = "";
            myPhone = "000-000-0000";
            isResdential = true;
            myElecRate = 0.12;
            myWaterRate = 2.35;
      }
```

```
// Constructor: This constructor initializes a utility customer to
// a specified setting. The following should be set as the customer
// myName should be set to the parameter name
// myPhone should be set to the parameter phone
// isResidential should be set to the parameter isRes
//
// If this is a residential customer:
//          myElecRate should be set to $0.12
//          myWaterRate should be set to $2.35
// If this is not a residential customer:
//          myElecRate should be set to $0.09
//          myWaterRate should be set to $3.15
public UtilityCustomer(String name, String phone, boolean isRes)
{
      myName = name;
      myPhone = phone;
      isResdential = isRes;

      if(isResdential) {
            myElecRate = 0.12;
            myWaterRate = 2.35;
      }
      else {
            myElecRate = 0.09;
            myWaterRate = 3.15;
      }
}

// This method returns the cost of the water bill.
// The water bill depends on whether it is a residential customer
// or a commercial customer according to the following:
// Residential:
//          $25.50 plus myWaterRate for each unit of water
// Commercial:
//          myWaterRate for each unit of water used.
public double waterBill(int waterUsed)
{
      if(isResdential) {
            return 25.50 + myWaterRate * waterUsed;
      }
      else {
            return myWaterRate * waterUsed;
      }
}

// This method returns the cost of the electric bill.
// The electric bill is computed the same whether it is a residential
// or commercial customer according to the following:
//          myElecRate for each unit of kwh used.
public double electricBill(double kwhUsed)
{
      return kwhUsed * myElecRate;
}

// This method returns the total bill, water plus electric.
// In order to receive full credit, you must invoke (use) the methods
// waterBill and electricBill. Rewriting the methods above will result
// in a loss of credit for this question.
public double totalBill(int waterUsed, double kwhUsed)
{
      return waterBill(waterUsed) + electricBill(kwhUsed);
}
```

```
	// Many autocaller phone systems do not know how to handle the dashes
	// in phone numbers, so it is required that the dashes are removed
	// from the phone number before passing it to the autocaller.
	//
	// This method returns the phone number of the utility customer
	// with the dashes (-) removed.
	//    Example: 707-433-5777 would be returned as 7074335777
	public String phoneNoDash()
	{
		return myPhone.substring(0, 3) +
			myPhone.substring(4, 7) +
			myPhone.substring(8);
	}
}
```

2. (15 points) This question asks you to implement a class to model a `StudentRecord`
The `StudentRecord` class should have the following:

- Data fields to hold the name, the grade level, the number of credits completed (decimal number), and the total grade points (decimal number)
- A constructor that takes a name and a grade level as parameters and initializes those fields. The other fields should be initialized to 0.
- A method that calculates and returns the GPA as a decimal number. GPA is calculated by dividing the total grade points by the credits completed.
- A method to calculate and return the credits needed to graduate. This method takes one parameter, the total number of credits required to graduate. You need to calculate and return the number of credits that the student needs. If the student already has the required number of credits, this method should return 0.
- A method to return the grade level of this student.

Please complete the class below. Use appropriate names for all fields and parameters.

```
public class StudentRecord
{
	private String myName;
	private int myGradeLevel;
	private double myCredits;
	private double myGradePoints;

	public StudentRecord(String name, int gradeLevel) {
		myName = name;
		myGradeLevel = gradeLevel;
		myCredits = 0.0;
		myGradePoints = 0.0;
	}

	public double GPA() {
		return myGradePoints/myCredits;
	}

	public creditsNeeded(double creditsRequired) {
		if(myCredits >= creditsRequired) {
			return 0;
		}
		else {
			return creditsRequired - myCredits;
		}
	}

	public int gradeLevel() {
		return myGradeLevel;
	}
}
```

JavaBat Type Questions (5 points each)

Logic

1. Complete the following method to determine if a triangle is acute, right, or obtuse. The rules to determine are as follows:

- if the sum of the squares of the first two sides is greater than the square of the third, the triangle is acute and triangleType should return 1.
- if the sum of the squares of the first two sides is equal to the square of the third, the triangle is right and triangleType should return 2.
- If the sum of the squares of the first two sides is less than the square of the third, the the triangle is obtuse and triangleType should return 3.
- If the side lengths could not make a triangle (a+b<=c), then return -1.

Note: you may assume that the side lengths are given in ascending order (smallest to largest)

triangleType(6, 8, 9) → 1
triangleType(6, 8, 10) → 2
triangleType(6, 8, 12) → 3
triangleType(6, 8, 14) → -1 (because the 3rd side is the same as the sum of the other two)
triangleType(6, 8, 20) → -1 (because the 3rd side is too long to make a triangle)

```
public int triangleType(int a, int b, int c)
{
      if(a + b <= c) {
            return -1;
      }
      int a2 = Math.pow(a, 2);
      int b2 = Math.pow(b, 2);
      int c2 = Math.pow(c, 2);

      if(a2 + b2 > c2) {
            return 1;
      }
      else if(a2 + b2 == c2) {
            return 2;
      }
      else {
            return 3;
      }
}
```

2. Complete the following method to compute the positive difference of the two parameters, a and b.

positiveDifference(9, 13) → 4
positiveDifference(13, 9) → 4
positiveDifference(-10, 0) → 10

```
public int positiveDifference(int a, int b)
{
   return Math.abs(a-b);
}
```

String

3. Complete the following method. This method is used to take a name in the format First Last (i.e. Mike Efram) and either return the first name or the last name depending on the value of the boolean parameter `returnFirst`. You may assume that the format of name is always 2 words with a space between the words.

nameParts("Mike Efram", true) → "Mike"
nameParts("Mike Efram", false) → "Efram"

```
public String nameParts(String name, boolean returnFirst)
{
	if(returnFirst) {
		return name.substring(0, name.indexOf(" "));
	}
	else {
		return name.substring(name.indexOf(" ")+1);
	}
}
```

(Extra Credit)

Recursion or Loops **(This method can be done recursively or with loops to receive credit.)**

4. Complete the following method.
This method returns the sum of all of the even digits in the nonnegative integer n.
Note: `n%2` can help determine if a number is even. `n%10` will give the last digit as a result. `n/10` will strip off the last digit.

sumEvens(0) → 0
sumEvens(13579) → 0
sumEvens(2468) → 20
sumEvens(12182008) → 20

```
public int sumEvens(int n)
{
	int sum = 0;

	while(n>0) {
		if(n%2 == 0) {
			sum += n%10;
		}
		n = n/10;
	}
	return sum;
}
```