Requirements

Each question carries 2 points.

- Deadline 22 sept, Friday, 11:59PM.
- Extra credit points will be added if total points are less than 10.
- Try every question and keep practicing from other online sites.
- Contact us if any assistance is needed.

1. Write a java function to calculate the salary of an employee based on the following rules.
   i. function takes input of number of hours an employee worked and returns the salary.
   ii. The first 36 hours worked are paid at a rate of 15.0, then the next 5 hours are paid at a rate of 15 * 1.5. Hours after that up to a max of 48 are paid at a rate of 15 * 2.

   Here is the prototype you can work with:

```
public double employeeSalary( double hours){
 // add your code here
 }
```

2. Write a java function that adds all the digits of an integer until it is single digit.
   i. function takes an integer as input and returns its sum of digits.
   ii. for example input = 37, sum = 3+7 = 10, sum = 1+0 = 1. result = 1.

   Here is the prototype you can work with:

```
public int addDigits( int input){

}
```

3. Write a java function to print all perfect number between 1 and n.

   i. Perfect number is a positive integer which is equal to the sum of its proper positive divisors.

   ii. For example: 6 is the first perfect number, Proper divisors of 6 are 1, 2, 3. Sum of its proper divisors = 1 + 2 + 3 = 6.

   Here is the prototype you can work with:

```
public void printPerfectNumbers( int n){

}
```

4. Write a java class called pizza with (Add detail as needed):

   i. At least 3 attributes :pizza type , unit price and loyalty points. More attributes are welcome to have.

   ii. Constructor is needed. Extra-credit for 0.5 point if you write more than 1 right constructor for this class
5. Write a java class called customer (Add detail as needed) :

   i. Attributes needed: customer name and what pizzas customer has ordered.
   ii. Think about what kind of data structure will be used to record the pizza name and numbers for each kind of pizza.(Give me your thought, Extra credit for 1 point)

   iii. In main method , sum up how many the customer spend.

EXTRA CREDIT

6. Write a Java program that generates an isosceles right angled triangle made of asterisks.

   i. function should take input of one equal side as integer. Other than the edges the inner part of the triangle should be empty.

   ii. For example input is 6. the function should print—

```
    *
    **
    * *
    *  *
    *   *
    ******
```

Here is the prototype you can work with:

```
public void printIsoscelesTriangle( int n){

}
```

Accomplishments

1. Calculate Employee Salary

Write a java function to calculate the salary of an employee based on the following rules.

i. function takes input of number of hours an employee worked and returns the salary.
ii. The first 36 hours worked are paid at a rate of 15.0, then the next 5 hours are paid at a rate of 15 * 1.5. Hours after that up to a max of 48 are paid at a rate of 15 * 2.

Analysis

This is an easy problem, so the code could explain itself.

Solution

```java
public double employeeSalary(double hours) {
    if (hours <= 0)
        return 0;

    double salary = 0;
    if (hours > 48) {
        // max salary
        salary += 36 * 15;
        salary += 5 * 15 * 1.5;
        salary += (48 - 36 - 5) * 15 * 2;
    } else if (hours > 41) {
        salary += 36 * 15;
        salary += 5 * 15 * 1.5;
        salary += (hours - 36 - 5) * 15 * 2;
    } else if (hours > 36) {
        salary += 36 * 15;
        salary += (hours - 36) * 15 * 1.5;
    } else {
        salary += hours * 15;
    }
    return salary;
}
```

2. Sum Digits

Write a java function that adds all the digits of an integer until it is single digit.

i. function takes an integer as input and returns its sum of digits.
ii. for example input = 37, sum = 3+7 = 10, sum = 1+0 = 1. result = 1.

Analysis

Use `%10` to get the least significant in an integer, and then set the integer to `/10`. Repeat this process until integer is 0.

Pre-Processing

However, the number `input` could be a negetive integer. In such case, we need to set it to its absolution value before processing. Otherwise, the sum could wrongly end up with a negative value.

Solution

```
    public int addDigits(int input) {
        if (input < 0) input = -input;
        int sum = 0;
        while (input != 0) {
            System.out.println(input % 10);
            sum += input % 10;
            input /= 10;
        }
        return sum;
    }
```

3. Perfect Numbers

Write a java function to print all perfect number between 1 and n.

i. Perfect number is a positive integer which is equal to the sum of its proper positive divisors *except itself*. ii. For example: 6 is the first perfect number, Proper divisors of 6 are 1, 2, 3. Sum of its proper divisors (*except itself*) = 1 + 2 + 3 = 6.

Analysis

A number i is a divisor of integer m only if `m%i` equals to `0`. So by iterating all the number in [1, m], we can find all the divisors of m, adding all of which (including itself) will get a sum. The number is a **perfect number** is the `sum = 2 * m`. In fact, we only need to iterate half of the numbers in [1,m] since `i` and `m%i` are symmetrical. We can create a help function `isPerfectNumber()` to find all the perfect numbers between 1 and n. Here is the solution.

Solution

```
public void printPerfectNumbers(int n){
  for(int m=1; m<=n; m++){
      if(isPerfectNumber(m)){
          System.out.println(m);
      }
  }
}

  private boolean isPerfectNumber(int num) {
      if (num == 0)
          return false;

      int sum = 0;
      for (int i = 1; i < Math.sqrt(num); i++) {
          if (num % i == 0)
              sum += num / i + i;
      }
      return sum - num == num ? true : false;
  }
```

4. Pizza

Write a java class called pizza with (Add detail as needed):

i. At least 3 attributes :pizza type , unit price and loyalty points. More attributes are welcome to have.
ii. Constructor is needed. Extra-credit for 0.5 point if you write more than 1 right constructor for this class

Analysis

- Add an extra attribute `name`
- Add 3 constructors

Solution

```
class Pizza{
    String type, name;
    double price;
    int loyaltyPoint;

    Pizza(){}
    Pizza(String name) {
        this.name = name;
    }
    Pizza(String name, double price){
        this.name = name;
        this.price = price;
    }
}
```

5. Customer

Write a java class called customer (Add detail as needed) :

i. Attributes needed: customer name and what pizzas customer has ordered. ii. Think about what kind of data structure will be used to record the pizza name and numbers for each kind of pizza.(Give me your thought, Extra credit for 1 point)
iii. In main method , sum up how many the customer spend.

Analysis

The attributes for this **Customer** class are: name, gender, age and the pizzas he/she orders.

In order to records the pizza name, we can use a collection of instances of class **Pizza** since it has an attribute *name*. However, it desn't have the attribute to record the number of each kind of pizza this customer orders. So we can add another attribute *number* in the class of **Pizza** to achieve this purpose.

To figure out how much this customer spends, we can simple calculate the money he/she spents in each kind of pizza and then add them together. Here comes the solution.

Solution

```
class Customer {
    String name;
    boolean gender;
    int age;
    Pizza pizzas[];

    public static void main(String[] args) {

        Customer bin = new Customer();

        // bin starts to order two kinds of pizzas
        bin.pizzas = new Pizza[2];
        bin.pizzas[0] = new Pizza("Summer Fire", 10.5, 1);
        bin.pizzas[1] = new Pizza("Winter Snow", 12.8, 2);

        // sum up the money bin spends
        double bill = 0;
        for (Pizza pizza : bin.pizzas) {
            bill += pizza.price * pizza.number;
        }

        System.out.println(bill);
    }
}
```

Also, here is the updated version of class **Pizza** definiation.

```
class Pizza{
    String type, name;
    double price;
    int loyaltyPoint;
    int number; // for 5. Customer

    Pizza(){}
    Pizza(String name) {
        this.name = name;
    }
    Pizza(String name, double price){
        this.name = name;
        this.price = price;
    }
    Pizza(String name, double price, int number){
        this(name, price);
        this.number = number;
    }
}
```

Note: A new constructor `Pizza(String name, double price, int number)` is added to record the number of pizza a customer orders.

6. Generates Isosceles Right Angled Triangles

Write a Java program that generates an isosceles right angled triangle made of asterisks.
i. function should take input of one equal side as integer. Other than the edges the inner part of the triangle should be empty.
ii. For example input is 6. the function should print—

```
    *
    **
    * *
    *  *
    *   *
    ******
```

Analysis

The first line only has one asterisk.

All the other lines except the last line have two asterisks, one of which is in the start of the line and the other is in the end.

The last line has as many asterisks as the line number itself.

Solution

```
    public void printIsoscelesTriangle(int n) {
        if (n <= 0) return;
        if (n == 1) {
            System.out.println("*");
            return;
        }
        // first line
        StringBuilder lines = new StringBuilder("*");

        // following lines
        for (int i = 1; i < n - 1; i++) {
            lines.append("\n*");
            for (int j = 1; j < i; j++)
                lines.append(" ");
            lines.append("*");
        }
        lines.append("\n");

        // last line
        while (n-- > 0) {
            lines.append("*");
        }

        // print all lines in the end
        System.out.println(lines);
    }
```

Note:

- Printing in the console is time consuming so I use StringBuilder cache all the output and print it at last.
- String "\n" in a StringBuilder means start a new line.

References

[1] ..\src\Assignment2.java