**SUTD 50.001 Introduction to Information Systems and Programming**

**Problem Set 1B (New)**

|  |
| --- |
| **Note:**   * **For all questions, please access the vocareum link found at eDimension for the starter code and to submit.** * **The Vocareum link is for submission only. Please work on the problems in Android studio, and this includes writing code for the test cases.** * **To prevent hard-coding, test cases used in Vocareum *may* be different from those provided here and will not be given to you.** |

**Question 1: Encapsulation [10 points]**

Create a class named **Person**.

* Implement the class with the following private attributes:
  + **name** (type String)
  + **age** (type int)
  + **gender** (type char)
  + **sharingConsent** (type bool).
* Provide constructor that takes 4 input arguments of name, age, gender, and sharing consent. The inputs are used to initiate the attributes of the same name.
* Provide public getter and setter methods to access and modify **age** attribute.
* Assume that the value of age passed to the Constructor and the setter for age will always be positive.
* Provide public getter for name and **sharingConsent** attributes
  + If **sharingConsent** is true, name getter will return the name, otherwise return a String “Anonymous”.

Additionally, create a class named **Filter**.

* implement a static method named **seniorFilter**. The method takes an array of **Person** objects and return an ArrayList of names whose age equals to or above 60 years old.

Reminder: For an attribute named foo, by convention the getter and setter is named getFoo() and setFoo().

Test Case:

import java.util.ArrayList;  
  
public class FilterPersonExample {  
 public static void main(String[] args) {  
 Person p1 = new Person("A", 90, 'F', false);  
 Person p2 = new Person("B", 60, 'M', true);  
 Person p3 = new Person("C", 30, 'F', true);  
 Person[] p = {p1, p2, p3};  
  
 System.*out*.println( Filter.*seniorFilter*(p) );  
 }  
}

Expected Output:

[Anonymous, B]

­­­­

* **Question 2: Polymorphism [10 points]**
* Create a class **Animal** with a method **makeSound**. There is no need to define a constructor. This method takes no input and return a String object “I am just an animal” .
* Create 3 **subclasses**: **Dog**, **Cat**, and **Cow**. Each subclass should override the **makeSound()** method to represent the unique sound that each animal makes. The String each subclass return for Dog, Cat, and Cow are “Woof”, “Meow”, and “Moo”.
* Create a subclass of **Cat** named **SiberianCat**. The class does not contain any method.
* Create a class called **AnimalConcert** that has a static method named **performConcert**.
  + The **performConcert** method should take an array of Animal objects as a parameter and return a String object as its output. The String is the sound of each animal joined with a comma and a space “, “.
  + So, if the input argument is an array consists of a Dog and a Cat object, the output of performConcert method is “Woof, Meow”

Test case:

public class TestSound {  
 public static void main(String[] args) {  
  
 Animal[] animals = {new Dog(), new Cat(), new Cow(), new SiberianCat()};  
  
 System.*out*.println( AnimalConcert.*performConcert*(animals) );  
  
 }  
}

Expected Output:

Woof, Meow, Moo, Meow

**Question 3: BaseInteger [30 points]**

**Overview**

An integer of arbitrary base can be represented as follows.

* "1,14,4" in Base 17 would be calculated in decimal value as:
* "43,5" in Base 60 would be:

In this problem, we assume that the base is an integer and 2 ≤ base ≤ 60, and the integer represented is 0 or positive.

Binary numbers and Hexadecimal numbers apply this same concept

* A binary number “110” will be represented as “1,1,0” in Base 2 with decimal value 6
* A hexadecimal number “1F9A” will be represented as “1,15,9,10” in Base 16 with decimal value 8090

The class BaseInteger stores information about such a positive integer with an arbitrary base.

**[20 points] Constructor and associated methods**

The constructor takes in two inputs:

* representation - a string with digits separated by commas and no spaces eg. "22,43,5"
* base - a positive integer from 2 to 60

These are assigned to the instance variables representation and base respectively.

The constructor is written for you.

Assumptions:

* representation will have at most four positions:
  + “X” – 1 position
  + “X,X” – 2 positions
  + “X,X,X” – 3 positions
  + “X,X,X,X” – 4 positions
* The number in each position will always be smaller than the base. There will be no test cases like this:

base = 3

representation = “1,2,3,0”

there cannot be ‘3’ as the largest possible digit in base 3 is 2

**Instance Variables and private methods**

Two private methods are called from the constructor to initialize the other instance variables, digits and decimalValue. Complete these private methods.

**convertRepresentationToArray()**

convertRepresentationToArray()reads the information from representation and stores the digits as elements in the digits int array.

Example:

if representation = “22,43,5”, after executing convertRepresentationToArray(), digits instance variable will contain {22, 43, 5}

**setDecimalValue()**

setDecimalValue()then reads the information in digits ,calculates the decimal value, and assigns the result to the instance variable decimalValue.   
 Example: if base = 60 and digits = {22, 43, 5}, after executing setDecimalValue(), decimalValue will be 81785.

**Getters**

To enable our autograder to check your output, implement the following getters:

* getDecimalValue()- returns the value of the instance variable decimalValue
* getDigitsString()- returns the String representation of the instance variable digits obtained by calling Arrays.toString()
* toString()- returns a string in the format "[representation] Base [base]" Eg:

"22,43,5 Base 60" where representation = “22,43,5” and base = 60

Do not write any setters other than setDecimalValue().

**[10 points] add(BaseInteger other, int base)**

add(BaseInteger other, int base) method takes in a BaseInteger object and a base value and returns a new BaseInteger object that is the sum, in base base, of the current object and other.

Here’s how it can be done:

1. Find the sum of the decimal values of this and other
2. Convert the sum to the new base base, and create the representation string. An example of how to do so is given here:

<https://www.tutorialspoint.com/computer_logical_organization/number_system_conversion.htm>   
Hint: you will need to calculate and store the individual digits in an array, reverse the order of the elements, and create the representation string.

1. Instantiate a new BaseInteger object which represents the result and return it

To help you in your implementation, two private helper methods are defined for you to complete. Since these methods are private, they will not be tested directly by the autograder. Thus, you are free to use them or to define and use your own private methods. Please remember to remove any print statements after you are done testing to keep the code clean.

private String deleteSpaces( String representation) takes in a String and returns a new string with the spaces removed

Example. “1, 2, 3, 4 ” --> “1,2,3,4”

private String convertBase( int decimalValue, int base) takes in a positive integer decimal value, with a base value, and returns the string representation of decimalValue in the new base   
Example. convertBase( 256, 16) --> “1,0,0”

Test Case:

public class TestBaseInteger {  
 public static void main(String[] args) {  
  
 BaseInteger baseInteger1 = new BaseInteger( "1,1", 17);  
 System.*out*.println(baseInteger1.getDecimalValue());  
 System.*out*.println(baseInteger1.getDigitsString());  
 System.*out*.println(baseInteger1);  
  
 BaseInteger baseInteger2 = new BaseInteger( "1,1", 60);  
 BaseInteger baseInteger3 = baseInteger1.add( baseInteger2 , 16);  
 System.*out*.println(baseInteger3.getDigitsString()); // [4, 15]  
 System.*out*.println(baseInteger3.getDecimalValue()); // 79  
 }  
  
}

Expected Output:

18  
[1, 1]   
1,1 Base 17  
[4, 15]  
79

**Testing**

At this point you should be familiar with writing your own tests and know how to instantiate the class object and the methods for testing. If you are unsure how to do testing, please read through this and go through the slides and start implementing tests for your older problem sets.

Here is an example of how you can test your class. Please also write your own additional test codes. This example shows how you instantiate a BaseInteger object and call the class methods to check if the values are correct by printing to console.

Our package import may be different. Do not blindly copy and paste.

Helps to make the test class in the same directory that contains your BaseInteger class.

package BaseIntegerPset;  
  
public class TestBaseInteger {  
 public static void main(String[] args) {  
 // Instantiate test1, decimal value 24  
 BaseInteger test1 = new BaseInteger("2,4", 10);  
 System.*out*.println("Test convertRepresentationToArray() and getDigitsString: ");  
 System.*out*.println(test1.getDigitsString());  
 System.*out*.println("Test setDecimalValue() and getDecimalValue(): ");  
 System.*out*.println(test1.getDecimalValue());  
 System.*out*.println("Test toString() ");  
 System.*out*.println(test1);  
 // Instantiate test2, decimal value 232 to test add(),  
 BaseInteger test2 = new BaseInteger("2,3,2", 10);  
 // add test1 with test 2 without changing base, store this into a new   
 // BaseInteger object ans1. ans1 should give you decimal value 256  
 BaseInteger ans1 = test1.add(test2,10);  
 System.*out*.println("Test add(), decimal value of ans1 is: ");  
 System.*out*.println(ans1.getDecimalValue());  
 // add test1 with test2 changing it to base 16, store this into a new

// BaseInteger object ans2  
 BaseInteger ans2 = test1.add(test2,16);  
 System.*out*.println("Test add() and change base, ans2 is: ");  
 System.*out*.println(ans2);  
 // ans1 and ans2 will have the same decimal value, just that they are   
 // represented by different bases.  
 System.*out*.println((ans1.getDecimalValue() == ans2.getDecimalValue()));  
  
 }  
}

**Output:**

Test convertRepresentationToArray() and getDigitsString:

[2, 4]

Test setDecimalValue() and getDecimalValue():

24

Test toString()

2,4 Base 10

Test add(), decimal value of ans1 is:

256

Test add() and change base, ans2 is:

1,0,0 Base 16

true

**Question 4: RSA Algorithm [35 points]**

**Overview**

RSA (Rivest–Shamir–Adleman) algorithm is one of the algorithms for data encryption. It involves 4 steps:

1. Public and Private key generation by a server. The key generation steps are:

* Choose 2 prime numbers **p** and **q**
* Compute modulus, ***n****=pq*
* Compute Carmichael function, ***l****=lcm(p-1, q-1)*, where *lcm* is the least common multiple. The formula of *lcm(a,b)* is |ab| / *gcd(a,b)*, where *gcd* is the greatest common divisor.
* Choose an integer **e** such that 2<e<l , gcd(e, l)=1, and e and l are coprime, their only common factor/divisor is 1.
* Calculate **d**, the modular multiplicative inverse of e modulo l. Private key consists of modulus **n** and **d**

1. Public key ***(n,e)*** distribution to a client
2. Data encryption by the client. Ciphertext ***c*** *= me mod n*. The value of *n* and *e* are the public key, and *m* is the message character as an integer.
3. Data decryption by the server. The integer character m can be recovered using the following formula, ***m*** *= cd mod n*. The value of *n* and *d* are private key, and *c* is the encrypted character in integer.

In this program, we will simulate a text message encryption and decryption. Server generates public key and private key. Public key is distributed to browser (client) and private key is kept secret by the server. Client can use the public key to encrypt a message while server can decrypt the message using the private key.

There are 2 public classes that you must complete.

**1. class Server**

1. Complete the **generatePublicPrivateKey** method. There are 6 TODOs here:
   1. Compute modulus ***n***
   2. Compute lambda ***l***
   3. Compute ***e*** using **computE** method.
   4. Compute ***d*** using **computeModInverse** method.
   5. Set ***(n,e)*** as the **publicKey** attribute. **publicKey** is an integer array of size 2. First element is the modulus **n** and the second is ***e***
   6. Set ***(n,d)*** as the **privateKey** attribute. **privateKey** is an integer array of size 2. First element is the modulus **n** and the second is **d**
2. Completethe **lcm** (least common multiple) and **gcd** (greatest common divisor) methods.
3. Complete the **decryptMessage** method. This method has BigInteger array as the input parameter. Each element of the BigInteger array represents the encrypted character. Each character needs to be decrypted and then its ASCII value is converted to a character. Concatenate all characters to output a String, which is the original message from the client (Browser).

**2. class Browser**

1. Complete the **establishConnection** method. This method has 1 input parameter: a Server object. There are 2 TODOs here:
   1. Invoke **generatePublicPrivateKey** of the Server object.
   2. Get the publicKey from the server and set it as the Browser object’s publicKeyattribute.
2. Complete the **encryptMessage**method. This method has 1 input parameter: a String object to be encrypted. The output of this method is a BigInteger array, where **each item represents the encrypted character** of the input message.   
   Hint: You can loop through each character of the String message, cast it to integer (*you can convert character into its ASCII value in Java by casting char to int*), and then using modPow method from BigInteger object to encrypt the character (*read on BigInteger documentation:* [*https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html*](https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html)*)*

All data fields, getter, setter, method signatures, isPrime method, as well as computE and computeModInverse are provided in the starter code. Do not change the provided code. You can add other methods if necessary.

Test Case:

import java.math.BigInteger;  
import java.util.Arrays;  
  
public class EncryptDecryptSimulation {  
 public static void main(String[] args) {  
 String msg = "red is sus";  
 Browser b = new Browser();  
 Server s = new Server();  
  
 s.setP(97);  
 s.setQ(53);  
 b.establishConnection(s);  
  
 BigInteger[] encryptedMsg = b.encryptMessage(msg);  
 String decryptedMessage = s.decryptMessage(encryptedMsg);  
  
 System.out.println(Arrays.*toString*(s.getPublicKey()));  
 System.out.println(Arrays.*toString*(s.getPrivateKey()));  
 System.out.println(Arrays.*toString*(encryptedMsg));  
 System.out.println(decryptedMessage);  
 }  
}

Expected Output:

[5141, 1237]  
[5141, 1021]  
[1284, 2324, 2103, 4631, 370, 396, 4631, 396, 515, 396]  
red is sus