Homework 1

Due Sep 2, 2020

What should be turned in?

* This word doc
* homework1 project (contains problem1, 2, 3, 4) in zip
* Problem5 project (Spring Boot project) in zip

# Problem 1: Procedural Programming vs OOP

**You need to write this program in the “problem1” package of the project I provided.**

Assume there is a **users.txt** which stores data about our users. The format of each row is

name&age&gender, e.g. Harry Potter&34&Male. I wrote a program in C to read every row from this file, convert it to a new format: name\tage\tgender (\t is a tab) and then sort all rows by age in increasing order, write back to a file called formatted\_users.txt.

|  |
| --- |
| struct User {  char name[64];  int age;  char gender[16];  };  struct User parse\_to\_user(char\* text) {  // parse “Harry Potter&34&Male” into a struct User  }  char\* format\_to\_text(struct User user) {  // convert a struct User to text like “Harry Potter\t34\tMale”  }  void sort\_users\_by\_age(struct User users[]) {  // TODO  }  void format\_user\_file(char\* origin\_file\_path, char\* new\_file\_path) {  // open files here...  struct User users[1024]; // Assume there are at most 1024 rows in this file  int count = 0;  while(…) { // read a line until the file is empty  struct User user = parse\_to\_user(line);  users[count++] = user;  }    sort\_users\_by\_age(users);    // create a new file based on new\_file\_path  for (int i = 0; i < count; ++i) {  char\* formatted\_user\_text = format\_to\_text(users[i]);  // write to new file...  }  // close files here…  }  int main(char\*\* args, int argv) {  format\_user\_file("users.txt", "formatted\_users.txt");  } |

The above code is written in a procedural style. Please use object-oriented style to implement the same program in Java. You can create any number of classes.

# Problem 2: Inheritance

**You need to write the program in the “problem2” package of the project I provided. There is no need to the methods in this problem, just write a System.out.println(“……”) inside the method, e.g. System.out.println(“Woof!”); System.out.println(“Driving!”);)**

Assume you work for a game company. You are asked to create dogs that can bark and cats that can meow. Because nature calls, they need to poop. So, you better prepare both class for that. Remember to use inheritance if you can.

Please copy your code here:

|  |
| --- |
| public class Animal {   public void Poop(String animal) {  System.*out*.println(animal + " is pooping.");  }   public void makeSound() {  System.*out*.println("Animal makes sound");  } }  public class Dog extends Animal {   @Override  public void makeSound() {  System.*out*.println("Woof");  } }  public class Cat extends Animal {   @Override  public void makeSound() {  System.*out*.println("Meow");  } } |

We have a lot of animals pooping everywhere, so we need a cleaning robot that can drive and clean. We also need a murder robot that can drive and kill all the cats and dogs (Thank goddess, this is just a game…)

Please copy your code about robots here (no need to implement drive, clean and kill, just print a message):

|  |
| --- |
| public class Robot {   public void Drive(String robot) {  System.*out*.print(robot + " is driving");  } }  public class MurderRobot extends Robot {   public void Kill() {  System.*out*.println("Killing animals");  } }  public class CleanRobot extends Robot {   public void Clean() {  System.*out*.println("Cleaning up poop.");  } } |

After a couple of months, your game becomes one of the most popular games on Steam! Your design of the inheritance hierarchy looks elegant, and you can reuse a lot of code by adopting inheritance. The product manager comes to you and says, our client demands a murder robot dog that can kill, drive and bark. But it does not have a digestive system, so it cannot poop.

After modifying your code, please copy your entire code about animals and robots here:

|  |
| --- |
| public class MurderRobotDog extends MurderRobot {   public void Bark() {  System.*out*.println("Woof");  } } |

# Problem 3: Polymorphism

**You need to write the program in the “problem3” package of the project I provided. After you are done, you can use TestDynamicArray to verify its correctness.**

Please implement a dynamic integer array class using only Java array (Integer[] or int[]). The default capacity of the dynamic array is 5. When we try to add a sixth element, this dynamic array can automatically resize to 10 (double the previous capacity but keep exiting items of course). Your dynamic array class should provide some public methods: get element by index, get the size of the array (number of elements in this array), add a new integer, remove an integer by index.

Paste you code here:

|  |
| --- |
| public class DynamicArray {   private int size = 0;  private int[] dynamicArray = new int[5];   public int get(int index) {  return dynamicArray[index];  }   public void add(int value) {  if (size != 0 && size % 5 == 0) {  int[] tempArray = new int[size\*2];  for (int i = 0; i < size; i++) {  tempArray[i] = dynamicArray[i];  }  dynamicArray = tempArray;  }  dynamicArray[size++] = value;  return;  }   public int size() {  return size;  }   public void delete(int index) {  for (int i = index; i < size; i++) {  dynamicArray[i] = dynamicArray[i+1];  }  size--;  } } |

Now that you have a dynamic integer array, please implement a more specific dynamic array called SortedDynamicArray. This array will make sure integers stored in the array are ALWAYS sorted (increasing order).

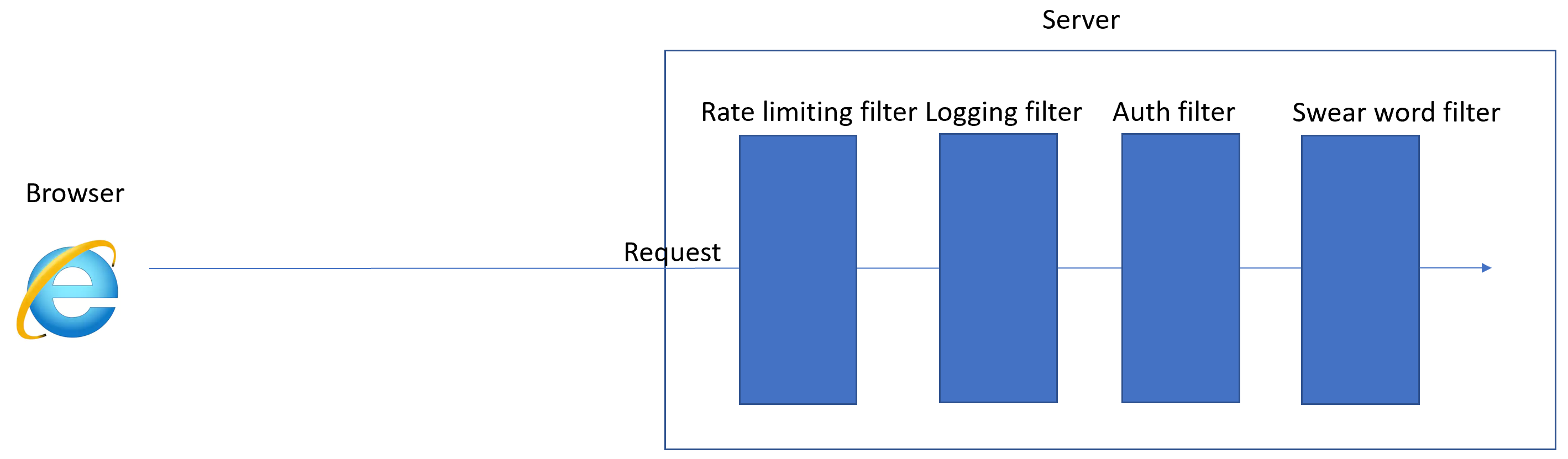
|  |
| --- |
| package problem3;  import java.util.Arrays;  public class SortedDynamicArray {   private int size = 0;  private int[] dynamicArray = new int[5];   public int get(int index) {  return dynamicArray[index];  }   public void add(int value) {  if (size != 0 && size % 5 == 0) {  int[] tempArray = new int[size\*2];  for (int i = 0; i < size; i++) {  tempArray[i] = dynamicArray[i];  }  dynamicArray = tempArray;  }  dynamicArray[size++] = value;  for (int i = 1; i < size; i++) {  int cur = dynamicArray[i];  int j = i - 1;  while (j >= 0 && cur < dynamicArray[j]) {  dynamicArray[j+1] = dynamicArray[j];  j--;  }  dynamicArray[j+1] = cur;  }  return;  }   public int size() {  return size;  }   public void delete(int index) {  for (int i = index; i < size; i++) {  dynamicArray[i] = dynamicArray[i+1];  }  size--;  } } |

# Problem 4: Interface

**You need to write the program in the “problem4” package of the project I provided. After you are done, you can use Client to verify its correctness. Same as problem 2, there is no need to implement the method in each filter. But you do need to print a message to indicate the method gets invoked.**

In a web application, a client (e.g. a browser) can send a request to the server application. Let us assume this request includes the IP address of the client, username and password, and a textual content (e.g. “Today is a great day, I am going to play basketball in the Rec center.”)

When the request arrives at the server side, the server needs a list of filters to examine it. Take a look at the picture.



Some common tasks that we can do with filters are:

1. Rate limiting: if requests with the same IP address appears over and over in a small period of time (say over 200 requests/minute), we need to limit this IP’s traffic, so discard this request immediately, terminate the filter chain.
2. Logging this request to log files
3. Authentication of this request (username and password)
4. Scanning the content of the request and automatically changing or censoring particular words or phrases based on a blocklist.

Your job is to define four filters that can achieve the above functionalities. Same as problem 2, you don’t need to implement the details of the functionalities, just print out a message indicating this function is called.

# Problem 5: Three-Tiered Architecture and Spring Boot

Please first watch the first 20 episodes of this link and then start working on the problem.

<https://www.youtube.com/watch?v=VO-1gF7jpzI&list=PLqq9AhcMm2oPW8xCPQeFoC0MBKuZFwahL>

Assume you work for an eBay-like website. This website allows people to buy/sell products from/to each other. This website creates an e-Wallet for every registered user. To purchase products on this website, a user must first buy the virtual currency offered by this website (aka credits, 1 credit = $1). Every e-Wallet has a unique id, a credit balance, a creation time (when this wallet is first created), a balance last modified time, and a status (normal or frozen).

Here are some requirements about this wallet:

1. User can view the current balance of their wallet.
2. The system shall increase the balance by a certain amount after the user buys credits.
3. When user A makes payment to user B, the system shall decrease the balance of user A and then increase the balance of user B. Note, the two operations must be done within the same transaction.
4. The system shall decrease the balance by a certain amount when the user withdraws the credit\*.
5. User can freeze the wallet if they want to. When a wallet is frozen, no balance modification is allowed.
6. User can unfreeze the frozen wallet.
7. Both id and creation time shall not be modified once they are initialized when the wallet is created. You should not expose interface to modify the two attributes.
8. Credit balance shall only be modified through the increase or decrease interface. No one can modify its value directly. When balance is modified, balance last modified time shall be updated as well.

\* “withdraw credits” means the credits are converted back to dollars (1 credit = $1) and transferred back to the user’s bank account. You do not need to implement about this step in this homework. We assume some third-party library is available.

Please implement this in Spring Boot based on what you learned in the video. If you encounter any problem, please feel free to ask me. You need to submit this Spring Boot project with homework1 project to TCU Online.

e-Wallet Application

Before testing using only API calls, please see the UI at localhost:8080!

API

POST

/api/create

Create a new wallet

PUT

/api/deposit/{walletID}

Query Params

("value", amount to deposit)

Deposit credits to wallet

PUT

/api/withdraw/{walletID}

Query Params

("value, amount to withdraw)

Withdraw credits from wallet

PUT

/api/transfer/{walletID}

Query Params

("value", amount to transfer)

("id2", receiver's walletID)

Transfer credits from one wallet to another

POST

/api/freeze/{walletID}

Freeze the wallet

POST

/api/unfreeze/{walletID}

Unfreeze the wallet

GET

/api/balance/{walletID}

Get the current balance of the wallet.