Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

The program is written in Java, it is designed to safely store passwords and to reliably encrypt messages which may then be delivered to others (method of delivery decided by the sender). The video shows the functions of the program work with test inputs. The "Encrypt" button took the text in the text field, ran it through encryption and displayed it in a new window. "Decrypt" did the same but backwards, taking in gibberish and turning it into readable English. "Generate" would generate new values for all characters and would require relaunching the program, so that's why I didn't run it.

2b)

The program was developed exclusively by me, Danylo Mirin, with some advice from a neighboring programmer who helped fix a problem during testing and with some code borrowed from Oracle demos on layouts. Initially, I thought to make some simple application that'd use a little window to encrypt and decrypt messages. However, it quickly grew into something bigger and now it is a password manager and an encryptor in one. Password manager worked flawlessly from the start but encryption had some reading issues, which were quickly resolved (this is where I asked a student next to me to look at the code and see where my testing method was going aloof). I used a lot of code that could throw exceptions and so I took the opportunity and made the ErrorLogger class, an abstraction of error handling. It is not the most interesting part of the project, however, the menu is. The menu is a grand combination of three smaller abstractions (encryption, password management, error logging) in an elegant and easy to use program

2c)

```
private void addActionListeners() {
    encrypt.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            String in = encinput.getText();
            if(in.equals("message...") || in.equals(""))
                JOptionPane.showMessageDialog(new JFrame(),"Enter something first!");
            else {
                if(!enc.empty()) {
                    JTextArea wat = new JTextArea(enc.encrypt(in));
                    JOptionPane.showMessageDialog(new JFrame(), wat);
                } else
                    JOptionPane.showMessageDialog(new JFrame(),
                    "No conversion file found. If you are supposed to receive a file from a friend,\nclose the program, add the file to the
                }
        }
    });
    decrypt.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            String in = encinput.getText();
            if(in.equals("message...") || in.equals(""))
                JOptionPane.showMessageDialog(new JFrame(),"Enter something first!");
            else {
                if(!enc.empty()) {
                    JTextArea wat = new JTextArea(enc.decrypt(in));
                    JOptionPane.showMessageDialog(new JFrame(), wat);
                } else
                    JOptionPane.showMessageDialog(new JFrame(),
                    "No conversion file found. If you are supposed to receive a file from a friend,\nclose the program, add the file to the
                }
        }
    });
    b.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            enc.generateNewValues();
            JOptionPane.showMessageDialog(new JFrame(), "Values generated.");
        }
    });
```

These are the most important action listeners from the Menu class. They run their code when the user wishes to encrypt or decrypt a message, thus effectively integrating the new encryption algorithm. That algorithm was designed specifically for this project but it can be reused later, in other projects. Unfortunately I wasn't able to capture the action listeners that handle password management in the same picture and I couldn't combine two either. However, what I can do it tell you how they work. Using JTextFields, they save user inputs and create a new account, which can then be found in the database. Database is saved permanently on a file and the program reads the data on the file each time the program is launched. Finally, there's a nice and simple GUI and that wraps the two new algorithms - encryption and password management into a bigger application.

2d)

```
/*#Encryption*/
public String encrypt(String message) {
    String scrambled = "";
    for(int i = 0; i < message.length(); i++)
        scrambled += alphabet.get(message.substring(i,i+1));
    return scrambled;
}
public void generateNewValues() {
    String[] chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()-_=+{}[]|;:'\",.<>/?".split("");
    for(String s : chars) {
        String value = keyGen();//generates a random 10-character key for encryption
        System.out.println(s+", "+value);
        alphabet.put(s, value);
    }
    try{
        writeToFile();
    } catch(IOException ex) {
        el.add(ex, 1);
    }
}
public String keyGen() {//In the EXTREMELY improbable event that keys are the same, just rerun the program. If that doesn't fix it...
    String[] chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890".split("");
    String key = "";
    for(int i = 0; i < 10; i++) {
        int rand = (int)(Math.random()*chars.length);
        key += chars[rand];
    }
    return key;
}

public String decrypt(String scrambled) {
    String message = "";
    //should always be false because every encryption value is 10 chars long
    if(scrambled.length() % 10 != 0)
        System.exit(5);
    for(int i = 0; i < scrambled.length(); i+= 10) {
        String expression = scrambled.substring(i,i+10);
        for(String s : alphabet.keySet())
            if(alphabet.get(s).equals(expression))
                message += s;
    }
    return message;
}
```

Here you can see a segment of code from the Encryptor class. It might not look like a lot but consider that there are about 50 more lines of code to maintain the class (read values from file, write them to file, error handling, etc). Now imagine rewriting the same ~100 lines of code every time I have to use encryption. That's abysmal, don't you think? Therefore it is necessary to have a class that will do all that behind the scenes and shows some simple to use methods that can be used anywhere else in the project. This I have done and that's why the code sample in 2c doesn't look too ugly (I could've written private helper methods to make action listneres neater). This is but a small program but even so, once the Encryptor class was complete and working, I never had to open it again to reread the lines of code and to reinterpret their function again. I just looked at method headers, checked parameters, and proceeded to code the action listners.