Panel 1

Last time

More String processing

Substitution operators

Worksheet on GUI program

Panel 2

```java
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;

public class Substituter extends JFrame implements ActionListener {
    private static final intEncode = 1;
    private static final int Decode = 2;
    private static final Font MEDIUM_FONT = new Font("Monospaced", Font.PLAIN, 17);
    private String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        .substring(0, 26) + "0123456789";
    private JTextField keyField = new JTextField(key);
    private JTextField inputField = new JTextField();
    private JButton encodeButton = new JButton("Encode");
    private JButton decodeButton = new JButton("Decode");

    public Substituter() {
        add(keyField);
        add(inputField);
        add(encodeButton);
        add(decodeButton);
    }

    public static String substitute(String str) {
        return str;
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == encodeButton) {
            String input = inputField.getText();
            System.out.println(input);
            // encode the input
            String encoded = substitute(input);
            inputField.setText(encoded);
        } else if (e.getSource() == decodeButton) {
            String input = inputField.getText();
            System.out.println(input);
            // decode the input
            String decoded = substitute(input);
            inputField.setText(decoded);
        }
    }
}
```
private void setKey(int node)
{
    try
    {
        String tapKey = keyField.getText();
        String tapLock = lockField.getText();
        if (tapKey.length() == tapLock.length())
            throw new Exception("" + tapKey.length());
        if (charAtNotShinPrime(tapKey))
            throw new Exception("" + tapKey);
        if (charAtNotShinPrime(tapLock))
            throw new Exception("" + tapLock);
        key = tapKey;
        lock = tapLock;
        if (node == ENCODE)
            encode();
        else if (node == DECODE)
            decode();
    }
    catch (Exception ex)
    {
        JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
    }
}

private void encode()
{
    String text = plainText.getText();
    if (text.equals(""))
        JOptionPane.showMessageDialog(this, "Try!");
    else
        codedText.setText(coded(text, lock, key));
}

private void decode()
{
    String text = plainText.getText();
    if (text.equals(""))
        JOptionPane.showMessageDialog(this, "Try!");
    else
        plainText.setText(coded(text, key, lock));
}

public void actionPerformed(ActionEvent ae)
{
    if (ae.getSource() == encodeButton)
        encode();
    else if (ae.getSource() == decodeButton)
        decode();
}

public static void main(String args[])
{
    Substitute p = new Substitute();
    p.setVisible(true);
}

Panel 3

Questions about Substituter (with GUI)

1) How many fields? List each field and its type.
   11 fields, etc.

3) How many methods? List each name, input, and return type.
   9 methods, etc.
Panel 5

c) Aside from the fact that only one method is static, which method is different from all the others and why?

   The 'Subclasser' method is different:
   - no return type
   - same name as class

   Such methods are called

   constructor

   and have special significance!

Panel 6

1) What is the difference between 'encode' and 'decode' method, and why do they really work?

   They work both, key

   for the 'code' method
Panel 7

e) There are several "??" in the code, indicating some error message. Replace the question marks by more appropriate Strings.

done

Panel 8

f) Explain in words what the method "charsAreNotUnique" does and how it works.

private boolean charsAreNotUnique(String s) {
    boolean duplicateFound = false;
    int i = 0;
    while (!duplicateFound && (i < s.length()-1)) {
        duplicateFound = (s.substring(i+1).indexOf(s.charAt(i)) >= 0);
        i++;
    }
    return duplicateFound;
}

i) ABCDE il name, returns -1
Panel 9

9) Explain how the “setKey” method works as best as you can.

Panel 10

10) Explain, if you can, when the “encode” method executes:

- Program starts => main executes
  - makes “new Subholder”
  - constructor executes
  - then prog. waits for an “action event”
- event is intercepted by
  - “action Performed”
  - and appra. action is performed.
Panel 11

Next topic: creating variables by the hundreds ...

Write a program that stores

\[ \text{1000 doubles} \]

\[
\begin{align*}
\text{double } x_1; \\
\text{double } x_2; \\
\vdots \\
\text{double } x_{1000};
\end{align*}
\]

\[ \Rightarrow \text{ use Array} \]

Panel 12

Declaring Arrays

An array is a sequential data structure that can store a fixed number of values of the same type.

Every Java array contains an additional variable named \texttt{length} to store the current size of the array. Arrays are declared using square brackets.

\[
\begin{align*}
\text{type arrayName}[\text{size}];
\end{align*}
\]

The default value of an array is \texttt{null}. Arrays are reference variables regardless of the types of their elements. They can be used as fields or local variables, or as input or output types of methods.

\[
\begin{align*}
\text{int } x; \\
\text{int } y[3];
\end{align*}
\]

\( x \) is a variable of type \texttt{int} \n\( y \) is an array of \texttt{int} variables (not yet initialized)
Panel 13

Initializing Arrays

To initialize an array that has previously been declared you use the keyword new to define an array of a specific size:

```java
arrayName = new typeName[size];
```

where size is an integer expression. You can declare and initialize arrays in one statement:

```java
type arrayName[] = new type[size];
```

1st - usual way

Individual array elements have their usual default values. You can also implicitly declare and initialize an array by listing its members in a comma-delimited list enclosed in curly brackets:

```java
type arrayName[] = { value1, value2, ..., valueN };  // sometimes
```

---

Ex: Declare 10 doubles. Declare 5 integers with values 10, 20, ..., 50

- double x1, x2, x3, x4, x5, x6, x7, x8, x9, x10;
- double x[] = new double[10];  // NEW AND NICE

```java
int i[] = {10, 20, 30, 40, 50};
```

Panel 14

Ex: Declare an array of 10 doubles and print out its values.

```java
double x[] = new double[10];
System.out.print(x);
```

- print out memory location where `x` is stored in RAM
- (not what you want)!

You need [ ] to access array elements

```java
for loops!!
```
Create an array of 1000 int, containing values 2, 4, 6, 8, 10, 12, ..., then print out the array.

```java
int x[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
for (int i = 0; i < x.length; i++)
    System.out.println(x[i]);

int x[] = new int[1000];
for (int i = 0; i < x.length; i++)
    x[i] = 2*(i+1);
for (int i = 0; i < x.length; i++)
    System.out.println(x[i]);
```