Panel 1

Last time:

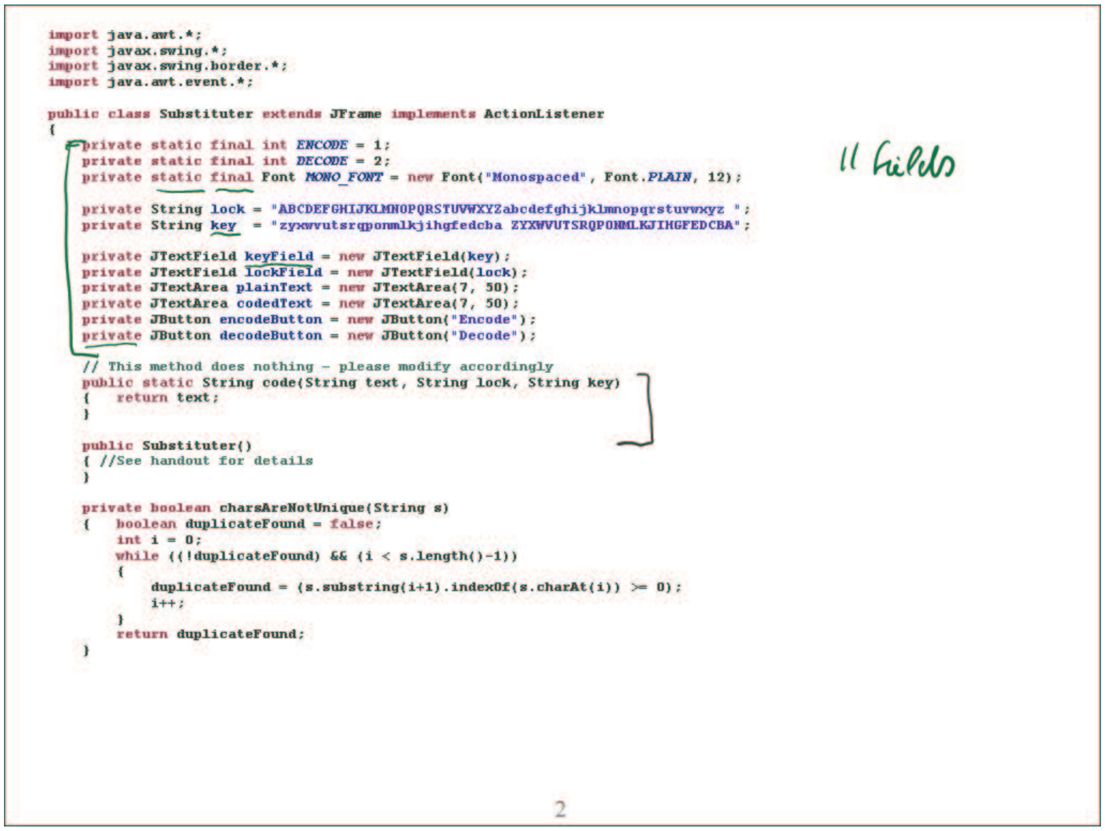
More String processing

Substitution cypher

Worksheet on GUI program

1

Panel 2

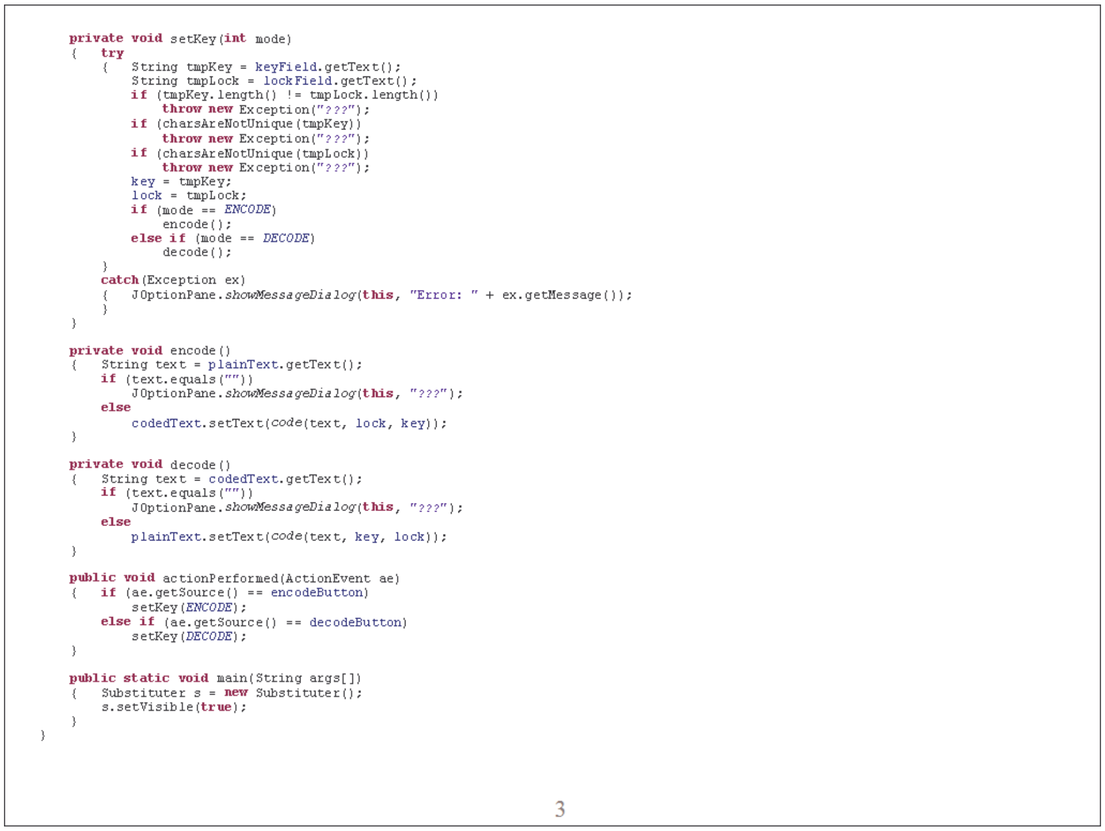


```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;

public class Substituter extends JFrame implements ActionListener
{
   private static final int ENCODE = 1;
   private static final int DECODE = 2;
   private static final Font MONO_FONT = new Font("Monospaced", Font.PLAIN, 12);

   private String lock = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz ";
   private String key  = "zyxwvutsrqponmlkjihgfedcba ZYXWVUTSRQPONMLKJIHGFEDCBA";

   private JTextField keyField = new JTextField(key);
   private JTextField lockField = new JTextField(lock);
   private JTextArea plainText = new JTextArea(7, 50);
   private JTextArea codedText = new JTextArea(7, 50);
   private JButton encodeButton = new JButton("Encode");
   private JButton decodeButton = new JButton("Decode");

   // This method does nothing - please modify accordingly
   public static String code(String text, String lock, String key)
   {   return text;
   }

   public Substituter()
   { //See handout for details
   }

   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;
   }
```

// fields

2

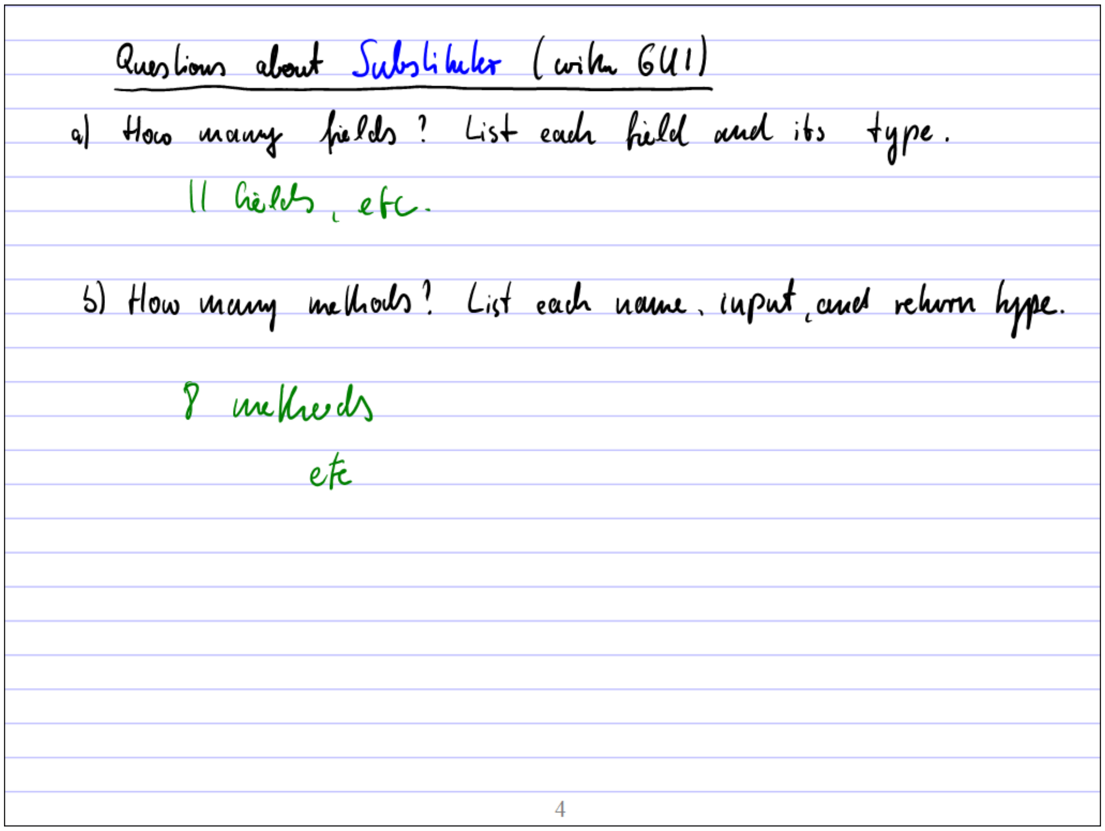
Panel 3



```
    private void setKey(int mode)
    {   try
        {   String tmpKey = keyField.getText();
            String tmpLock = lockField.getText();
            if (tmpKey.length() != tmpLock.length())
                throw new Exception("???");
            if (charsAreNotUnique(tmpKey))
                throw new Exception("???");
            if (charsAreNotUnique(tmpLock))
                throw new Exception("???");
            key = tmpKey;
            lock = tmpLock;
            if (mode == ENCODE)
                encode();
            else if (mode == DECODE)
                decode();
        }
        catch(Exception ex)
        {   JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

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

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

    public void actionPerformed(ActionEvent ae)
    {   if (ae.getSource() == encodeButton)
            setKey(ENCODE);
        else if (ae.getSource() == decodeButton)
            setKey(DECODE);
    }

    public static void main(String args[])
    {   Substituter s = new Substituter();
        s.setVisible(true);
    }
}
```

3

Panel 4



Questions about Substituter (with GUI)

a) How many fields? List each field and its type.

11 fields, etc.

b) How many methods? List each name, input, and return type.

8 methods

etc

4

Panel 5

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

The 'Substituter' method is different:

- no return type
- same name as class

Such methods are called

constructor

and have special significance!

5

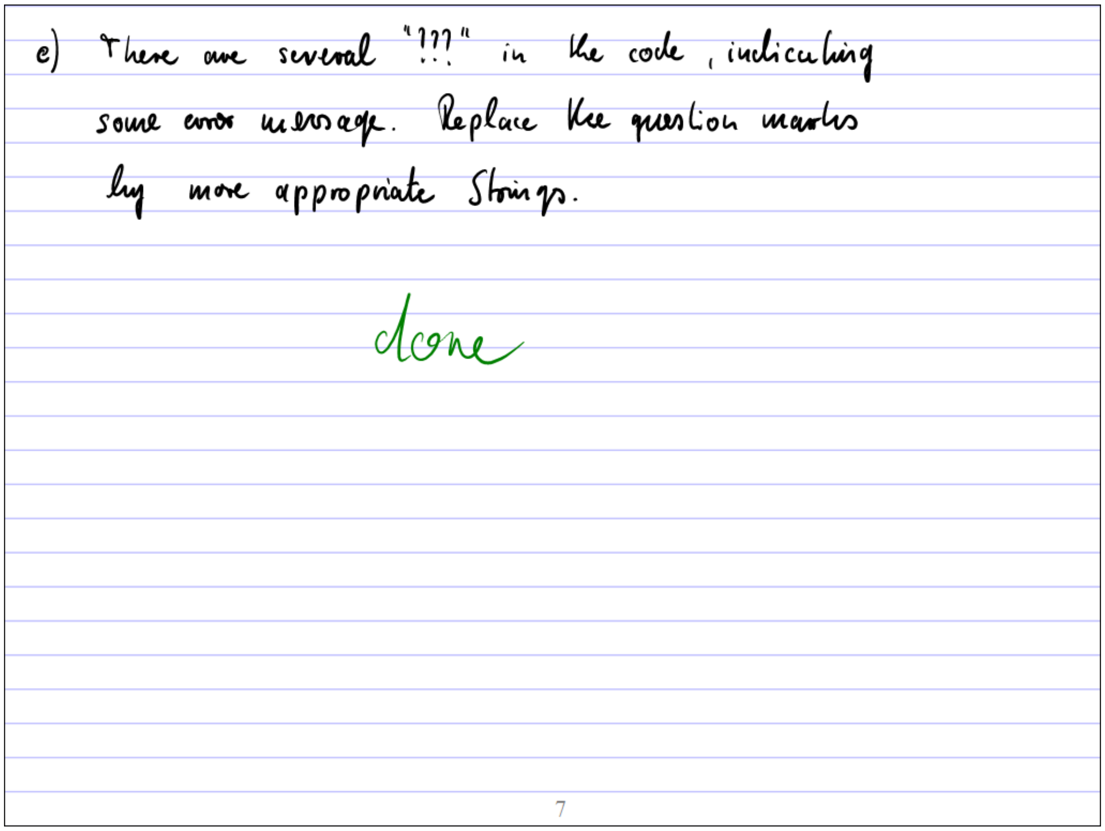
Panel 6

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

they switch lock, key for the "code" method

6

Panel 7

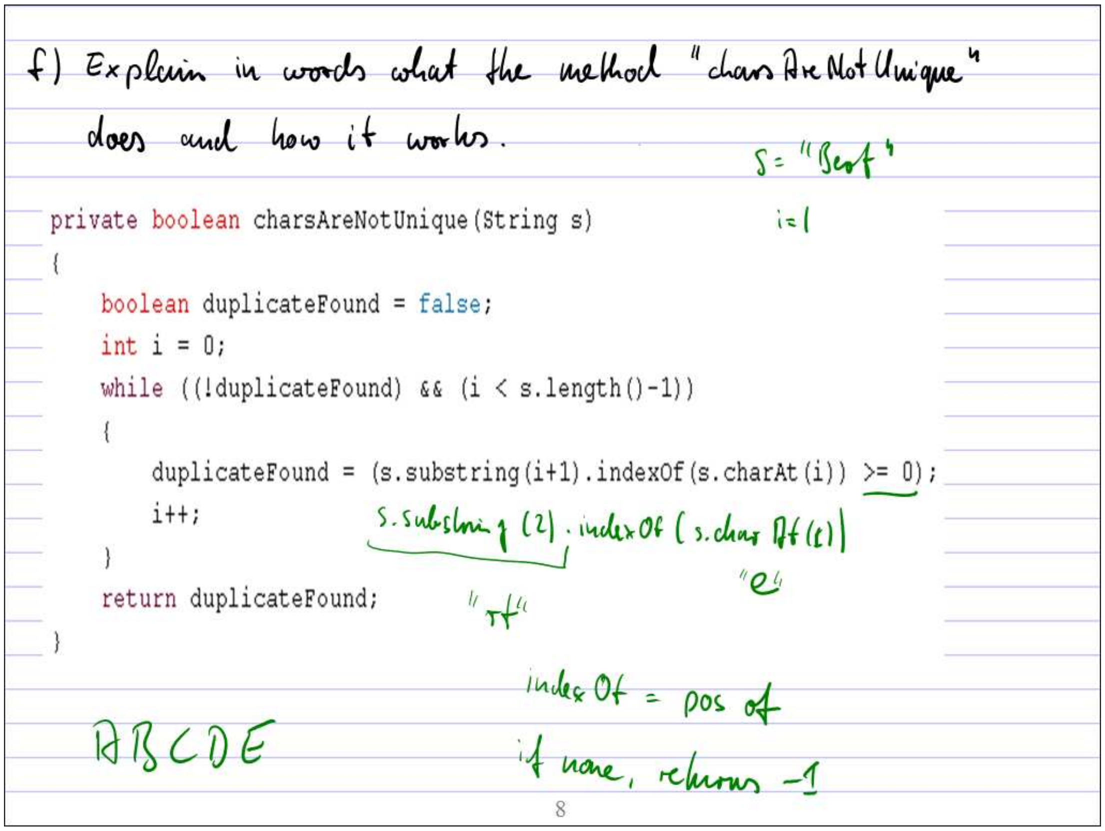


e) There are several "???" in the code, indicating some error message. Replace the question marks by more appropriate Strings.

done

7

Panel 8



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

S = "Bert"

i = 1

```
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;
}
```

$\underbrace{\text{s.substring(2)}}_{\text{"rt"}}\text{.indexOf(}\underbrace{\text{s.charAt(1)}}_{\text{"e"}}\text{)}$

indexOf = pos of

if none, returns -1

ABCDE

8

Panel 9

g) Explain how the "setKey" method works as best as you can

9

Panel 10

h) Explain, if you can, when the "encode" method executes

Program starts ⇒ main executes,

makes "new Substituter"

→ constructor executes

then prog. waits for an "action event"

event is intercepted by

"actionPerformed"

and appr. action is performed!

10

Panel 11

Next topic: creating variables by the hundreds ...

Write a program that stores 1000 doubles

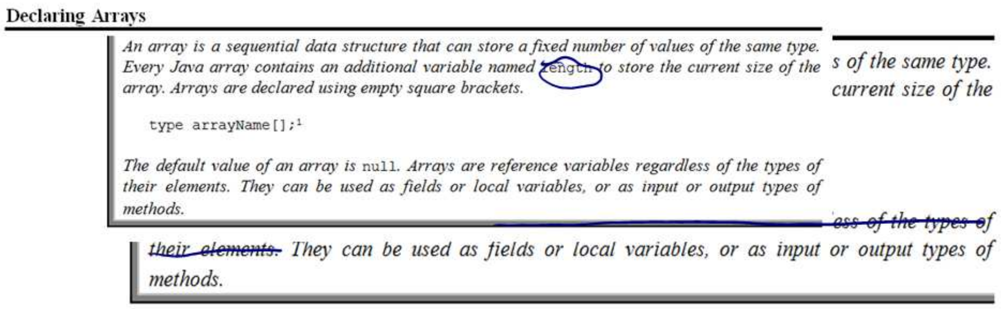
```
double x1;
double x2;
   ⋮
double x1000;
```

} too much work

$\Rightarrow$ use Array

11

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* `length` *to store the current size of the array. Arrays are declared using empty square brackets.*

```
type arrayName[];¹
```

*The default value of an array is* `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.*

*s of the same type. current size of the*

*~~ess of the types of their elements.~~ They can be used as fields or local variables, or as input or output types of methods.*

int x; x is a variable of type int

int y[]; y is an array of int variables (not yet initialized)

12

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:*

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

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

```
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:*

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

sometimes

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

(A) double x1, x2, x3, x4, x5, x6, x7, x8, x9, x10; BAD

(B) double x[] = new double[10]; NEW AND NICE

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

Panel 14

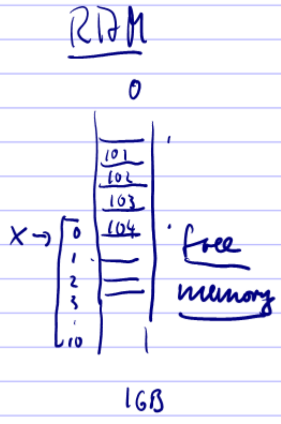
Ex: Declare an array of 10 doubles and print out its values.

in BlueJ

double x[] = new double[10];

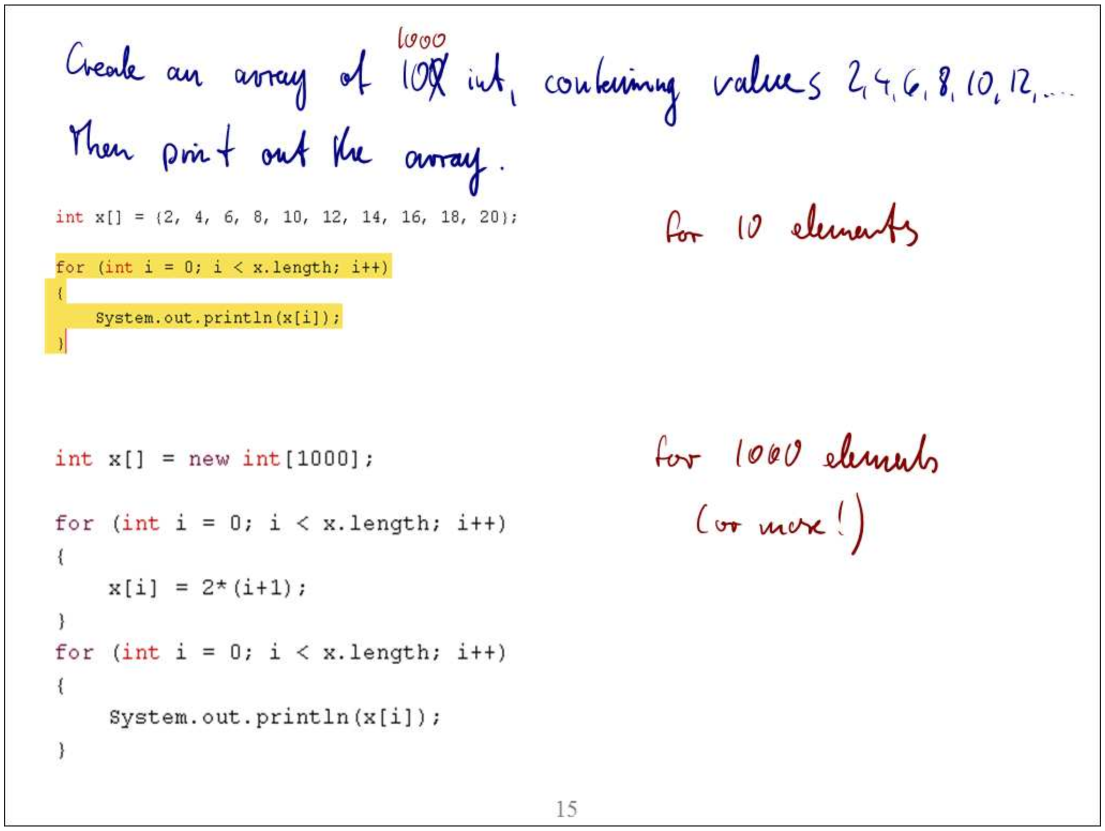
System.out.println(x);

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



You need [ ] to access array elements + for loops !!!

Panel 15



Create an array of $\overset{1000}{\cancel{10}}$ int, containing values 2, 4, 6, 8, 10, 12, ...
Then print out the array.

```
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]);
}
```

for 10 elements

```
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]);
}
```

for 1000 elements (or more!)

15