|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IMPORTS**  Import java.io.\*; //input output obvs  Import java.util.\*; //date, time, rand  import javax.swing.\*;//for jframe, jbutton, etc import javax.awt.\*;//for layouts import java.awt.event.\*; //for events and button functions | **System**  System.out.println();  System.out.print();\\no newline  If start with string, if you want to add numbers, need to be in ()  If start with numbers, then they will compile   |  |  |  | | --- | --- | --- | | System.out.println("println (1+2) = " + (1+2)); | System.out.println(1+2); | System.out.println("println 1 + 2= "+1+2); | | Println (1+2) = 3 | 3 | Println 1 + 2 = 12 | | **Scanner**  Scanner \_\_\_ = new Scanner(System.in) |
| **Exceptions**  Type function() throws IOException//won’t ask | **Data Types, Loops, Branches, Arrays** | **Windows** |
| **Interfaces and Nested Classes** | **Threads** |  |
| **ETC** |  |  |
| **HOMEWORK 1 - important stuff -Caesar Cipher**   |  |  |  | | --- | --- | --- | | import java.util.\*; //A = 65 Z = 90  import java.io.\*;  public class Assignment1\_AnVu {  static File getFilename(){  System.out.print("Enter your input filename: ");  Scanner sin = new Scanner(System.in);  String filename = sin.nextLine();  while(!new File(filename).canRead()){  System.out.println("Bad Filename");  System.out.println("Please enter your input filename: ");  filename = sin.nextLine();  }  File myFile = new File(filename);  return myFile;  }  static int getJob(){  System.out.println("What would you like to do?");  System.out.println("1 - Encrypt a file");  System.out.println("2 - Decrypt a file");  System.out.println("3 - Exit");  System.out.print("Your choice: ");  Scanner sin = new Scanner(System.in);  int job = sin.nextInt();  while(job < 1 || job > 3){  System.out.println("That ain't a choice!");  System.out.print("Your choice (1, 2, 3): ");  job = sin.nextInt();  }  return job;  }    static String getKey(){  System.out.print("Enter your key: ");  Scanner sin = new Scanner(System.in);  String key=sin.nextLine();  return key;  }  //based off of the difference between string lengths, the key length would be extended  static String extendKey(int extra, String key){  String extended = key;  while (extra > 0){  if (extra >= key.length()){  extended += key;  extra -= key.length();  }  else{  extended += key.substring(0,extra);  extra -= extra;  }  }  return extended;  } | static String matchCase(String original, String output){  StringBuilder send\_out = new StringBuilder();  System.out.println(original.length());  System.out.println(output.length());  for (int i = 0, n = original.length(); i < n;i++){  char orig = original.charAt(i);  char comp = output.charAt(i);  char change;  if (Character.isLetter(comp)){  if(Character.isLowerCase(orig)){  change = Character.toLowerCase(comp);  }  else{change = comp;}  send\_out.append(change);  }  else{send\_out.append (comp);}  }  return send\_out.toString();  }  static void writeToFile(String output, String output\_file)throws IOException{  PrintStream outFile = new PrintStream(output\_file);  outFile.println(output);  }  static char changeChar(char original,char key,boolean encrypt){  int changed\_int;  char changed;  System.out.println(((char)original )+": " +( (char)key));  System.out.println(((int)original )+": " +( (int)key));  if (encrypt == true){  changed\_int = original + key;  }  else{  changed\_int = original - key;  }  System.out.println(changed\_int);  while(changed\_int > 90 || changed\_int < 65){  if (changed\_int >=156){  changed\_int -= 91;  }  else if(changed\_int < 156 && changed\_int > 90){  changed\_int -= 65;  }  else if(changed\_int <65 && changed\_int > 0){  changed\_int += 65;  }  else{  changed\_int += 91;  }  } | changed = (char)changed\_int;  System.out.println(changed\_int +": "+changed);  return changed;  }  static void encrypt(File input, String output, String key)throws IOException{  Scanner inFile = new Scanner (input);  while(inFile.hasNextLine()){  String line = inFile.nextLine();  String upper = line.toUpperCase();  String upper\_key = key.toUpperCase();  if (key.length() < upper.length()){  int charCount = upper.length() - key.length();  upper\_key = extendKey(charCount,upper\_key);  }  String output\_line = "";  for (int i = 0, n = line.length(); i < n;i++){  char c = upper.charAt(i);  char out\_char;  if (Character.isLetter(c)){  out\_char = changeChar(c, upper\_key.charAt(i), true);  }  else{out\_char = upper.charAt(i);}  output\_line += out\_char;    }  String final\_out = matchCase(line,output\_line);  writeToFile(final\_out,output);  }  }    public static void main(String[] args) throws IOException{  // TODO code application logic here  int job = getJob();  while (job < 3){  String key = getKey();  File input = getFilename();  String output = getOutputFile();  if (job == 1){  encrypt(input,output,key);  }  else{  decrypt(input,output,key);  }  job = getJob();  }  System.exit(0);  }    } | | **HOMEWORK 2 - Color Buttons**   |  |  | | --- | --- | | import java.util.ArrayList;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  import java.util.\*;  public class ColorChanger {  static int num\_button = 12;  static JFrame jf;  static JPanel jp;  static JButton[] buttons;  static Random rand;  public static void createButtons(){  buttons = new JButton[num\_button];  rand = new Random();  for(int i = 0; i < num\_button; i++){  JButton butt = new JButton(); butt.addActionListener(new ButtonPress());  int r = rand.nextInt(255);  int g = rand.nextInt(255);  int b = rand.nextInt(255);  Color random = new Color(r,g,b); butt.setBackground(random);  buttons[i] = butt;  }  }  public static void createPanel(){  jp.setLayout(new GridLayout(3,4));  for(int i = 0; i < num\_button; i++){  jp.add(buttons[i]);  }  } | public static void main(String[] args) {  // TODO code application logic here  try{ UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());  } catch(Exception e){  e.printStackTrace();  }  jf = new JFrame("Change Other Colors");  jf.setSize(600,600);  jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  jp = new JPanel();  createButtons();  createPanel();  jf.add(jp);    jf.setVisible(true);  }  static class ButtonPress implements ActionListener{  @Override  public void actionPerformed(ActionEvent e) {  JButton bx = (JButton)e.getSource();  for(int i = 0; i < num\_button; i++){  if (bx != buttons[i]){  int r = rand.nextInt(255);  int g = rand.nextInt(255);  int b = rand.nextInt(255);  Color random = new Color(r,g,b); buttons[i].setBackground(random);  }  }  }  }  } | | **HOMEWORK 3 - Threaded Buttons**   |  |  | | --- | --- | | import java.util.ArrayList;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  import java.util.\*;  import java.lang.\*;  public class ThreadedButtons {  static int num\_button = 12;  static JFrame jf;  static JPanel jp;  static ArrayList<JButton> buttons;  static ArrayList<Boolean> pressed;  static Random rand;  public static void createButtons(){  rand = new Random();  for(int i = 0; i < num\_button; i++){  JButton butt = new JButton();  butt.addActionListener(new ColorChange());  int r = rand.nextInt(255);  int g = rand.nextInt(255);  int b = rand.nextInt(255);  Color random = new Color(r,g,b);  butt.setBackground(random);  buttons.add(butt);  pressed.add(false);  }  }  public static void createPanel(){  jp.setLayout(new GridLayout(3,4));  for(int i = 0; i < buttons.size(); i++){  jp.add(buttons.get(i));  }  }  public static void main(String[] args) {  // TODO code application logic here  //mac stuff  try{ UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());  } catch(Exception e){  e.printStackTrace();  }  buttons = new ArrayList<JButton>();  pressed = new | ArrayList<Boolean>();  jf = new JFrame("Change Colors all the times");  jf.setSize(600,600); jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  jp = new JPanel();  createButtons();  createPanel();  jf.add(jp);  ColorChange test = new ColorChange();  test.start();  jf.setVisible(true);  }    static class ColorChange extends Thread implements ActionListener{  public static long last = System.currentTimeMillis();  @Override  public void actionPerformed(ActionEvent e) {  JButton bx = (JButton)e.getSource();  int found = buttons.indexOf(bx);  if(pressed.get(found)) pressed.set(found,false);  else pressed.set(found,true);    }  public void run(){  while(true){  long current = System.currentTimeMillis();  if (current - last > 1000){  for(int i = 0; i < buttons.size(); i++){  int r = rand.nextInt(256);  int g = rand.nextInt(256);  int b = rand.nextInt(256);  Color random = new Color(r,g,b);  if(!pressed.get(i)) buttons.get(i).setBackground(random);  }  last = current;  }  }  }  }  } | |