Today’s topics

Java
Looping
Upcoming
Arrays in Java

Reading
Great Ideas, Chapter 3

Looping/Iteration/Repetition

- Much of power of computer comes from the ability to repeat
  - Can use “button pushing” for slow, controlled loop
  - Use language features for full-speed looping
- While-loop syntax
  While (logical expression)
  {
    statement;
    ...
    statement;
  }
- Repeat statements between braces as while logical expression is true

While statement

- Risk of infinite loop
  - Usually a serious error
  - Something in body of loop must alter logical expression
- Gauss summation
  int sum = 0;
  int k = 0;
  while (k < 100)
  {
    k = k + 1;
    sum = sum + k;
  }
- \( sum = \frac{n(n+1)}{2} \)

Compound Interest

- Redo our compound interest example
  - Specify how many months to compute loan for
  - Don’t require the push of a button for each month
- Code:
  public class CompInterest extends java.applet.Applet
  implements ActionListener
  {
    TextField mInstruct, mBalance;
    DoubleField gRate, gPrinc, gPay;
    Button bCompute;
    IntField gMonths;
    double rate, princ, pay, balance;
    int months, k;
### Compound Interest.2

```java
public void init()
{
    mInstruct = new TextField(80);
    mInstruct.setText("Enter principal, rate, payment, #months; then press 'Compute'");
    gPrinc = new DoubleField(10);
    gRate = new DoubleField(10);
    gPay = new DoubleField(10);
    gMonths = new IntField(10);
    bCompute = new Button("Compute");
    mBalance = new TextField(80);
    bCompute.addActionListener(this);
    add(mInstruct); add(gPrinc); add(gRate); add(gPay);
    add(gMonths); add(bCompute); add(mBalance);
}
```

### Compound Interest.3

```java
public void actionPerformed(ActionEvent event)
{
    Object cause = event.getSource();
    if (cause == bCompute)
    {
        princ = gPrinc.getDouble();
        rate = gRate.getDouble()/12;
        pay = gPay.getDouble();
        months = gMonths.getInt();
        balance = princ;
        k = 0;
        while (k < months)
        {
            balance = balance*(1.0 + rate) - pay;
            k = k + 1;
        }
        mBalance.setText("After " + months + " months at " +
                        100*rate*12 + "/\\% and payments of " + pay +
                        " the balance is " + balance);
    }
}
```

### Many uses for Loops

- Can count up or down
  - Previous example counts up, month by month
  - In class, showed decrementing from 10, by 1
- Don’t have to increment or decrement by 1
  - Can change by any value
  - E.g., for even number: start at 0, increment by 2
- Data dependent loop
  - Logical expression may depend on data
  - Increment may depend on data
  - Data input may provide halting value: called sentinel
- Whimsical example to draw a diamond

### Diamond Example

```java
public class Diamond extends java.applet.Applet implements ActionListener
{ TextField tf;
    TextArea ta;
    Button bDraw;
    String stars = "***************";
    String spaces = "          ";
    int k;
    public void init()
    {
        tf = new TextField("Hello    ");
        ta = new TextArea(22, 20);
        ta.setFont(new Font("Monospaced", Font.BOLD, 12));
        bDraw = new Button("Draw");
        bDraw.addActionListener(this);
        add(tf); add(bDraw); add(ta);
    }
```

Diamond Example.2

```java
public void actionPerformed(ActionEvent event)
{
    Object cause = event.getSource();

    if (cause == bDraw)
    {
        tf.setText("Goodbye");
        k = 0;
        while (k < 10)
        {
            ta.append(spaces.substring(0,10-k) +
                       stars.substring(0,2*k+1)+"\n");
            k = k + 1;
        }
    }
}
```

Diamond Example.3

```java
k = 1;
while (k < 10)
{
    ta.append(spaces.substring(0,1+k) +
              stars.substring(0,19-2*k)+"\n");
    k = k + 1;
}
```

- Contains many new things
  - String: substring
  - TextArea:setFont, append, "\n"