

Today's topics

Java

Writing Functions/Methods

Upcoming

Information Retrieval

Reading

Great Ideas, Chapter 4

Writing Functions/Methods

- **Function is Synonym of Method**
 - **Funcion is more generic term**
 - **Method is used in Java**

- **Syntax of a function**

```
return-type name(parameter-list)
{
    statement;
    . . .
    statement;
    return return-type-object // if return-type not void
}
```

- **Return-type**
 - **May be void if no information returned**
 - **If not void, type must match type of info returned**
 - **If not void, 1 or more *return statements* required**

Writing Functions/Methods

- **Parameter list**
 - **type-object pairs, separated by commas**
 - **Can be empty**
 - **Parameters used to obtain info from outside of function**
- **Return statement**
 - **Sends information back to statement that invoked function**
 - **Type must match return-type on function header**
- **Miscellaneous**
 - **Header may be preceded by `public` to make function available anywhere in program**

Use of Functions

- **Void Functions**

- **Use by putting on line with parameters (if any)**

- **E.g.,**

- `object-name.setText("Hello");`

- **Functions that return info**

- **Must use in calculation or assignment**

- **Include parameters (if any)**

- **E.g.,**

- `weight = object-name.getDouble();`

Using Functions

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

Using Functions.2

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

Using Functions.3

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

    if (cause == bDraw)
    {
        tf.setText("Goodbye");
        Dia();
        Mond();
    }
}
}
```

- **No New Capability**
 - **Packaged differently from Diamond.java code**

Using Parameters

- Now take advantage of parameters to set size of diamond

```
public class DiaParms extends java.applet.Applet
                               implements ActionListener

{ IntField gSize;
  TextArea ta;
  Button bDraw;
  String stars = "*****";
  String spaces = "          ";
  public void init()
  { gSize = new IntField(10);
    ta = new TextArea(20, 20);
    ta.setFont(new Font("Monospaced", Font.BOLD, 12));
    bDraw = new Button("Draw");
    bDraw.addActionListener(this);
    add(gSize); add(bDraw); add(ta);
  }
```

Using Parameters.2

```
void Dia(int sz) // this line changed
{ int k = 0;
  while (k < sz) // this line changed
  { ta.append(spaces.substring(0,10-k) +
              stars.substring(0,2*k+1)+"\n");

    k = k + 1;
  }
}

void Mond(int sz) // this line changed
{ int k = sz; // this line changed
  while (k > 0)
  { k = k - 1;
    ta.append(spaces.substring(0,10-k) +
              stars.substring(0,2*k+1)+"\n");
  }
}
```

Using Parameters.3

```
public void actionPerformed(ActionEvent event)
{
    int sz; // this line new
    Object cause = event.getSource();
    if (cause == bDraw)
    {
        sz = gSize.getInt(); // this line new
        Dia(sz); // this line changed
        Mond(sz-1); // this line changed
    }
}
}
```

- **Almost no difference in code for functions**
 - **Great difference in flexibility**

Partially Used Array

- Often array is sized for worst case
 - Only use the “front” part of it in most cases
 - Need variable to keep track of how much we use
 - Such variable often called *size* or *used*
- Saw an example in `ArrayStats` program:

```
double mean(double[] list, int size)
{
    int k = 0;
    double sum = 0.0;
    while (k < size)
    {
        sum = sum + list[k];
        k = k + 1;
    }
    return sum/size;
}
```

Functions that work on Arrays

- The previous was an example of using an array parameter
- Such functions that “work” on arrays are common
- Possible Applications:

- **Sum values**

```
double arraySum(double data[], int size);
```

- **Count number of occurrences of a specific value**

```
int arrayCountZero(double data[], int size);
```

```
int arrayCount(double data[], double val, int size)
```

- **Set all values to a constant value**

```
void arraySet(double data[], double val, int size);
```

- **Locate (search for) a particular value**

```
int arrayFind(double data[], double key, int size);
```

Information Retrieval

- Often want to use program for information storage and retrieval
- On line phone book is good example
- Using “Parallel” or “Corresponding Arrays”
 - Array (of Strings) for Names
 - Array (of Strings) for Phone Numbers

Name	Number
“J.Able”	“613-1978”
“D.Ramm”	“660-6532”
“D.Ramm”	“732-7616”
“R.Odom”	“681-6326”
“M.Salter”	“684-8111”
“W.Tars”	“613-1978”