CS1013 – Programming Project

Dr. Gavin Doherty
ORI LG.19
Gavin.Doherty@cs.tcd.ie

Trinity College Dublin

DSG
Distributed Systems Group

Simple text entry widget

```
class TextWidget extends Widget {
  int maxlen;

  TextWidget(int x,int y, int width, int height,
  String label, color widgetColor, PFont font, int event, int
   maxlen){
    this.x=x; this.y=y; this.width = width; this.height= height;
    this.label=label; this.event=event;
    this.widgetColor=widgetColor; this.widgetFont=font;
    labelColor=color(0); this.maxlen=maxlen;
  }
  void append(char s){
    if(s==BACKSPACE){
      if(!label.equals(""))
        label=label.substring(0,label.length()-1);
    }
    else if (label.length() <maxlen)
      label=label+str(s);
  }
}
```

Main program

```
ArrayList myWidgets;
TextWidget focus;
PFont stdFont;
static final int TEXT_WIDGET=1; static final int EVENT_NULL=0;

void setup(){
  stdFont=loadFont("Verdana-12.vlw");
  textFont(stdFont);

  TextWidget textedit=new TextWidget(100, 100, 100, 40,
  "edit me!", color(255), stdFont, TEXT_WIDGET, 10);
  TextWidget another=new TextWidget(100, 200, 100, 40,
  "no me!", color(255), stdFont, TEXT_WIDGET, 10);
  focus=null;
  myWidgets = new ArrayList();
  // Create two widgets, then add them to the myWidgets list.
  myWidgets.add(textedit);  myWidgets.add(another);
  size(400, 400);
}
void draw(){
  background(200);
  for(int i = 0; i < myWidgets.size(); i++){
    ((Widget)myWidgets.get(i)).draw();
  }
}
```

Input handling

```
void mousePressed(){
  int event;

  // Ask the widgets on the list if the current mouse value is
  // inside them. If it is, the widget has been pressed.
  // Take the appropriate response to this event.

   for(int i = 0; i < myWidgets.size(); i++){
    TextWidget theWidget = (TextWidget)myWidgets.get(i);
    event = theWidget.getEvent(mouseX,mouseY);
    if(event== TEXT_WIDGET) {
      println("clicked on a text entry widget!");
      focus= theWidget;
      break;
    }
    else {
      focus=null;
    }
  }
}
```

Handling keyboard input

```
void keyPressed(){
  if(focus != null) {
    focus.append(key);
  }
}
```

Recap – program outline

- Setup

```
read_in_the_file(); // done, week 1
result = default_query();
current_query = query3;// whatever type of query is default
```

- Draw

```
switch(current_query){
    case query1:
        render_query1(results); // done, week 2
        break;
    case query2:
        Etc…..
    }
render_controls();
```

Recap – input handling

- mousePressed()
  - Work out which button pressed

```
switch(event)
    case button 1:
        current_query = query1;
        results=query1();
        break;
    case button 2:
        current_query = query2;
        results=query2();
    Etc.
```

  - You may need several “results” variables for different types of data returned by different queries.

This week - queries

- Goal is to define a number of methods to process the data loaded into the csv file.
- Return transactions for a particular address, or a particular date.
- Highest priced transactions.
- Average(s) for a particular area.
- Good solution:
- As above, but sort by price/date/area.

Transactions for an address

- Pass two parameters – ArrayList of datapoints and String search field. Return list of datapoints (all will have same address).

```
ArrayList queryAddress(String address, ArrayList
  priceData)
```

- Create ArrayList for results.
- Go through priceData and check if address value is the same as the target address.
- If so, copy the data to the results ArrayList.

Highest priced transactions

```
ArrayList queryHighestPrice(ArrayList priceData)
```

- If we have less than the desired number of results, add the current transaction to the results list. Otherwise, check whether it is higher than the lowest item in the results list (Hint: for-loop inside a for-loop). If so, replace the lowest item in the results list, and calculate the new lowest item in the results list.

Average prices

- Pass in ArrayList of datapoints. Return ArrayList of class (e.g. averagePrice) containing area name, the total transactions for that area, and the number of properties for that area.

```
ArrayList queryAveragePrices(ArrayList
  priceData)
```

- For each datapoint, check results list if there is a result for that area, if there isn't then create one, and set `numberOfProperties` value to 1, and transactionTotals to the value of the transaction. If there is one, increase the value by 1, and add the transaction price to the total.

Demonstration Goals

1. Implement at least three different queries on the data.
2. Have some selection mechanism to invoke the different queries (this can be very basic – eg. press a key).
3. Draw the results of the queries on the screen.