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 EntryWidget extends Widget {
  int maxlen;

  EntryWidget(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;
EntryWidget focus;
PFont stdFont;
static final int ENTRY_WIDGET=1; static final int EVENT_NULL=0;

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

  EntryWidget textedit=new EntryWidget(100, 100, 100, 40,
  "edit me!", color(255), stdFont, ENTRY_WIDGET, 10);
  EntryWidget another=new EntryWidget(100, 200, 100, 40,
  "no me!", color(255), stdFont, ENTRY_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++){
    EntryWidget theWidget = (EntryWidget)myWidgets.get(i);
    event = theWidget.getEvent(mouseX,mouseY);
    if(event== ENTRY_WIDGET) {
      println("clicked on an 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 hotel ratings for user.
- return amount hotels have been rated
- return average ratings of hotels

- Good solution:
- As above, but sort by date, hotel name, times rated or average rating.

Hotels rated by user

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

```
ArrayList query_ratings_of_user(String username,
  ArrayList ratingData)
```

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

Most often rated hotels

- Pass in ArrayList of datapoints. Return ArrayList of class (e.g. RatedHotels) containing Hotel Name and an integer representing number of times rated.

```
ArrayList query_most_watched(ArrayList
  ratingData)
```

- For each datapoint, check results list if there is a result for that hotel, if there isn't then create one, and set `number_of_ratings` value to 1. If there is one, increase the value by 1. (Hint: for-loop inside a for-loop).

Average ratings for hotels

- Similar to most rated. One way to implement is for results to contain total number of ratings, plus total points for all those ratings. The average is obtained by dividing points by number of ratings:

```
ArrayList query_avg_rating(ArrayList hotelData)
```

- For each datapoint, check results list if there is a result for that hotel, if there isn't create one, and set `number_of_ratings` value to 1, and total points to value of this rating. If there is one, increase the value by 1, and add rating to `total_points`. (Hint: for-loop inside a for-loop).