Week 13

~~**h9** due before 10pm on **Monday 4/22 (w13)**~~
p6 available soon and due before 10pm on **Friday 5/3 (w14)**
(w14 - start in-class in and complete in week 14)

Team Project: QuizGenerator
milestone 1 design: due before 10 pm on Thursday 4/18 (w12 - our design available on 4/19)
milestone 1 GUI: due before 10pm on on Thursday 4/25 (w13)
milestone 3 final program: due before 10pm on Thursday 5/2 (w14)

Follow submission instructions to create `executable.jar` and `.zip`.
Submit the .zip file
See Milestone #3 rubric

Project Design Diagram is available via Canvas and:
https://pages.cs.wisc.edu/~deppeler/cs400/assignments/p5/files/design.pdf

Peer Mentors: JavaFX, JSON parsing, Streams
Zhiyue (W and F 10-12 in 1289) and Yiye (Th 12:30-2:30pm in 1358, F 9-11 in 1358)
Exam Review m 5/6 11am-1pm, 1289 cs.
Read: Module 13 (get started on Module 14 reading)

THIS WEEK:

- More JavaFX
  - Event Handling
  - Node, Alert, and Dialog
- Java 8 Streams
  - pipeline operators
  - terminal operators
- StreamsPractice
  - Examples: process word list using Java 8 Stream operations

Next Week

- HTML/CSS/JS
- Final Exam Topic Review
- Course Evaluations

javafx.scene UI Control: Object <- Node

JavaFX "super" type of layout managers and other UI controls
allows controls and layout managers to be placed within other managers

UI Control Example: Node <- ImageView

```
Image image = new Image("pretty_picture.jpg");
ImageView imageView = new ImageView(image);
borderLayout.setCenter(imageView);
```

- setMaxWidth
setFitWidth.

UI Control: Node <- Parent <- Region <- Control <- ListView

provide a clickable list that triggers events
add controls to layout manager, add layout manages to other layout
add to scene and stages

```
ListView<String> list = new ListView<String>();
ObservableList<String> items = FXCollections.observableArrayList (
    "First Item", "Second Item", "Third Item"
);
list.setItems(items);
list.setPrefWidth(100);
list.setPrefHeight(80);
```

Add UI Controls to VBox, HBox, FlowLayout layout managers

getChildren() - return a observablelist can add to

```
layoutManager.getChildren().add(item)
layoutManager.getChildren().addAll(item …)
layoutManager.getChildren().clear()
layoutManager.getChildren().remove(item)
layoutManager.getChildren().removeAll(item …)
layoutManager.getChildren().setAll(item …)
layoutManager.getChildren().size()
```

Example:

```
VBox bBox = new VBox();
vBox.getChildren().addAll( loginBox, imageView,
     new HBox( cancelButton, okButton ) );
```

Events

Events are generated when the user interacts with GUI, for example when user:

- enters text into a text field
- clicks a button (left or right-clicks a button or control)
- moves the mouse (hovers over a control) onEnter onExit
- click and drags a scroll bar
- clicks and drags a border control (make window bigger)
- types a key KeyListener

What can an event handler do when an event occurs?

TextField. getText

label. setText(" ")

add and remove controls

show or hide (enable / disable)

can creat stage (dialogue – model stage)

set scene and stage

Event Handling Examples

Label Event

event

front on hovers

```
label.setOnMouseEntered( e ->
       label.setStyle( "-fx-font-size: 20pt;" ); ) ;
```

TextField Event

```
nameInput.setOnAction( e ->
      vBox.getChildren().addAll(
           new Label( nameInput.getText() ) ) );
```

Button Event

```
button.setOnAction( e -> buttonAction(); );
```

Display an Alert Dialog Window when context menu is requested for a Label

right click

```
// Display alert dialog when context menu is requested for a label
nameLabel.setOnContextMenuRequested(
    event -> {
        Alert alert = new Alert( AlertType.INFORMATION, "Enter your first name" );
        alert.showAndWait().filter(
            response -> response == ButtonType.OK );
    }
);
```

filter expression

`.filter` is a ***stream*** function that only passes data thru if the data matches or passes the filter expression.

JavaFX Event Handling

a(n) Lambda expression does it all!

1. defines an unamed / anonymous instance
2. of an anonymous inner class (that implements EventHandler <ActionEvent>)
3. and becomes the rull/behavior for the action event
   registered handler

Display a modal Dialog Window

```
// Display a form dialog window (Stage) that can be closed and return to "owner" Stage
   GridPane form = ... // create layout manager with form fields
   Scene newScene = new Scene(form,600,200);
   final Stage dialog = new Stage();
   dialog.initModality(Modality.APPLICATION_MODAL);
   dialog.initOwner(primaryStage); ← owner
   dialog.setScene(newScene);
   dialog.show();
```

JavaFX and CSS

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-file/cssref.html

- contains rules for each style
- applied at runtime
- can change program styles without recompiling program
- similar to HTML css stylesheet syntax
- prepend: -fx-
- not required knowledge for cs400

```
scene.getStylesheets().add(getClass().getResource("application.css").
toExternalForm());
```

application.css

```
.scroll-pane .viewport {
     -fx-background-image: url("background.jpg");
}

.label {
    -fx-font-size: 12px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333; <- color (grey)                   transprency
                                                                  ↓
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}                           ↑              ff  ff  ff                ↑ shadow.
                           style

.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}

.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}

#welcome-text {
   -fx-font-size: 32px;
   -fx-font-family: "Arial Black";
   -fx-fill: #818181;
   -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 );
}

#actiontarget {
  -fx-fill: FIREBRICK;
  -fx-font-weight: bold;
  -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

Copyright 2018-9 Deb Deppeler, @2019 Andy Kuemmel

Java 8 Streams

http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

What? a conduit ("pipeline") from sourse data to final result.

Why? more intuitive for some problem
declare operations
works on "Big Data" - does not fit in memory
can be parallelized "easily" - must merge "reduce"

Requires three things

1. Data sources
2. A "chain" of intermediate operations
3. one Terminal operation - ends somewhere

input is a stream
output is a stream

Comparison with Collections

Collections	Streams
- fit in memory - store data - require external info - data can be reused DATA	- may not fit in memory - pipeline for data to flow through - may have internal iterator - data is consumed - supports functional programming - can be infinite - easy to parallelize. Behavior

db.stream()

Intermediate Operations form pipelines

.filter(Predicate)

[1,2,3,4,5,6,7,8,9,10] → predical .filter(v-> v % 2 == 0) → [2, 4, 6, 8, 10]

stream

.map(Function) y=f(x)

["cat", "In", "haT" is" , "bAck"] → .map(String::toLowerCase) → ["cat", "in", "hat", "is", "back"]

.distinct ← x dulplicate

["cat", "In", "haT" in" , "bAck"] → .map(String::toLowerCase).distinct → "cat" "in" "hat" "back"

.limit(n)

[3 , 5, 4, 7, 5, 2] → .sorted.limit(3) → 2, 3, 4 ? 3, 4, 5

.skip(n)

["cat", "In", "haT" in" , "bAck"] → .skip(2) → "haT", "in", "bAck".

Terminal operation

.findFirst *.findAny* *.findAll(* predicate

"cat" any of them all match requirement

short-circuiting

- finish when limit is reached
- answer is known, found first item
- the terminal produce a finite result in some cases - from a infinite resource

Other intermediate operations

source operation array => stream

.sort .iterate .generate Stream.of(1,2,3,4)

.mapToInt .mapToDouble .mapToLong

produce a specilized stream {int double long} .sum

Terminal (Aggregate) Operations

.reduce – must be associative as order is not-deterministic – repeat the operation until result is known

.collect (to list)

.sum .max .count
specialized stream

Intermediate
.sorted
.forEach – non-deterministic – Terminal
.~~forEachOrdered~~ – deterministic, not as efficient
.~~into~~ ~~.iterator~~
.~~iterator~~
.toArray() – returns an array of type Object
.~~anyMatch~~ – short-circuiting terminal – true if any match
.~~allMatch –~~ short-circuiting terminal – true if all match
.~~noneMatch –~~
.findFirst - short-circuiting terminal – returns first element
.findAny - short-circuiting terminal – returns any element

Example: Get Word List from a File

In Java 7: get word list from a file

```
String filepath = filename; // relative or absolute filename
List<String> wordList = new ArrayList<String>();
Scanner filescnr = new Scanner(new File(filepath);
while ( filescnr.hasNextLine() ) {
    String line = filescnr.nextLine();
    if ( line != null && ! line.equals(""))
        wordList.add(line.trim().toUpperCase());
}
```

In Java 8: get word list from a file

// try with-resources

```
try ( Stream<String> wordStream =
       Files.lines(Paths.get(filepath)) )
{
  return wordStream
    .map(String::trim)                         // trim whitespace
    .filter( x -> x != null && x != "" )       // keep non empty lines
    .map(String::toUpperCase);                 // convert to upper case

} catch (IOException e) {
    e.printStackTrace();
    return null;
}
```

redundent

StreamsPractice Examples

```
/p/course/cs400-deppeler/public/html-s/code/StreamsPractice_Andy
/p/course/cs400-deppeler/public/html-s/code/StreamsPractice_Deb

public class StreamsPractice {
  public static void main(String[] args) throws IOException {
    List<String> words = Arrays.asList(
      "the", "Quick", "Brown", "the", "THE",
      "fox", "jumped", "jUmped", "over", "the", "lAzy", "dog"
    );

    List<String> list = getSortedWordsList(words,3,2);
    System.out.println(list);

    Iterator<String> iter = printWordBlanks(words,"e");
  }

  // return sorted, lowercase, words with min length AS LIST
  private static List<String> getSortedWordsList(
    List<String> words, int minlength, int n) {
                           ↑ existing list can be added to → not used
    // collector is a list
    List<String> result = words.stream()
                 .map(thing -> thing.toLowerCase())
                 .sorted()
                 .distinct()
                 .filter(word -> word.length() >= minLength)
                 .limit(n)
                 .collect(Collectors.toList());
    return result;
  }

  // print match words with blanks for selected letter
  // caution: only words for single letter matches (as written)
  private static void printWordBlanks(
    List<String> words, String matchChar ) {
    words.stream()
         .map(x ->x.toLowerCase())
         .distinct()
         .filter(n -> n.contains(matchChar))
         .forEach(thing -> {
             for (int i=0; i < thing.length(); i++){
                 if (matchChar.contains(thing.charAt(i)+""))
                    System.out.print(thing.charAt(i)+" ");
                 else System.out.print("_ ");
             }
             System.out.println();
          }
    );
// see code online for more examples
```