COMP 504: Graduate Object-Oriented Programming and Design

Lecture 1: Intro, Union Design Pattern, Spark Java

Mack Joyner (mjoyner@rice.edu)

https://www.clear.rice.edu/comp504

Administration

Class: MWF 9:45-10:40am (online only)

Zoom Link: https://riceuniversity.zoom.us/j/91084163450?pwd=RHJUWnh0RlNtMmNBd1Vvc2RUQk9yZz09

TAs: Ying Zhou, Yunda Jia

Instructor: Mack Joyner (DH 2063)

Office Hours: See course website (www.clear.rice.edu/comp504)

Piazza: https://piazza.com/class/k40c8v1awpov6

Grades: Canvas

Github classroom: hw1: https://classroom.github.com/a/6smfT_ne

Acknowledgment: *Special thanks to Scott Pollack and Stephen Wong*

Git Classroom

- We'll use GitHub classroom
  - Create a GitHub account if you don't have one
  - Click on GitHub repo link (slide 2)
  - Link git id (net id) to git repo (first repo)
- Submit code often
  - Recommend using an IDE (IntelliJ - ultimate edition)
  - git add
  - git commit -m "description of changes"
  - git push origin master (don't forget)

Course Goals and Expectations

- Learn good software design principles
- Deliver effective oral design and project presentations
- Learn how to work effectively on team projects
- Class participation, in-class exercises (5% of grade) due before start of next class
- Have fun

Industrial Goal: Marketable Software Engineer

- Rapid prototyping in popular programming language: https://www.youtube.com/watch?v=Og847HVwRSI
- Building modular code
- Encapsulation
- Robust software
- Unit testing
- Design Reviews
- Team projects involving members with different team roles
- Oral presentation, communicate technical solutions to peers that and inform and demonstrate completion of project requirements

Design Process

- Separate invariant code from variant code
  - Invariant functionality (abstract classes, interfaces)
  - Variant functionality in concrete classes
- Implement invariant aspect of the problem once
  - Reduces errors
- Multiple instances may exist of variant parts
- Create solution by combining
  - customized variant component(s)
  - single invariant component

Model-View-Controller (MVC)

- Model contains the data
- View presents the data to the user in response to the Model
- Controller sends commands to the update the Model or the View

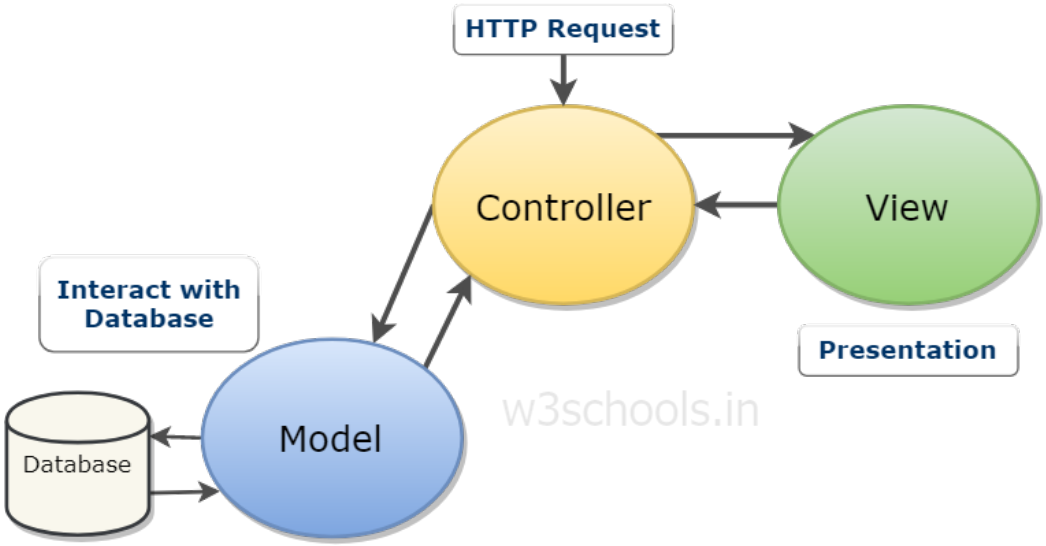
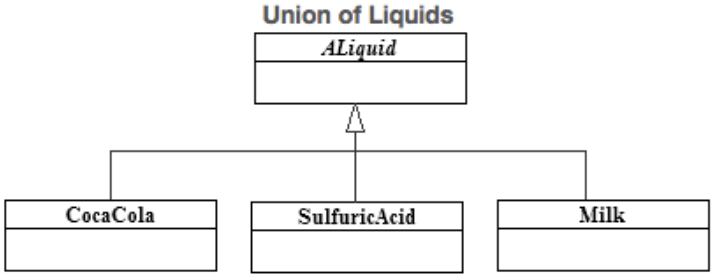


Fig: MVC Architecture

Source: https://www.w3schools.in/mvc-architecture/

Union Design Pattern (Model)

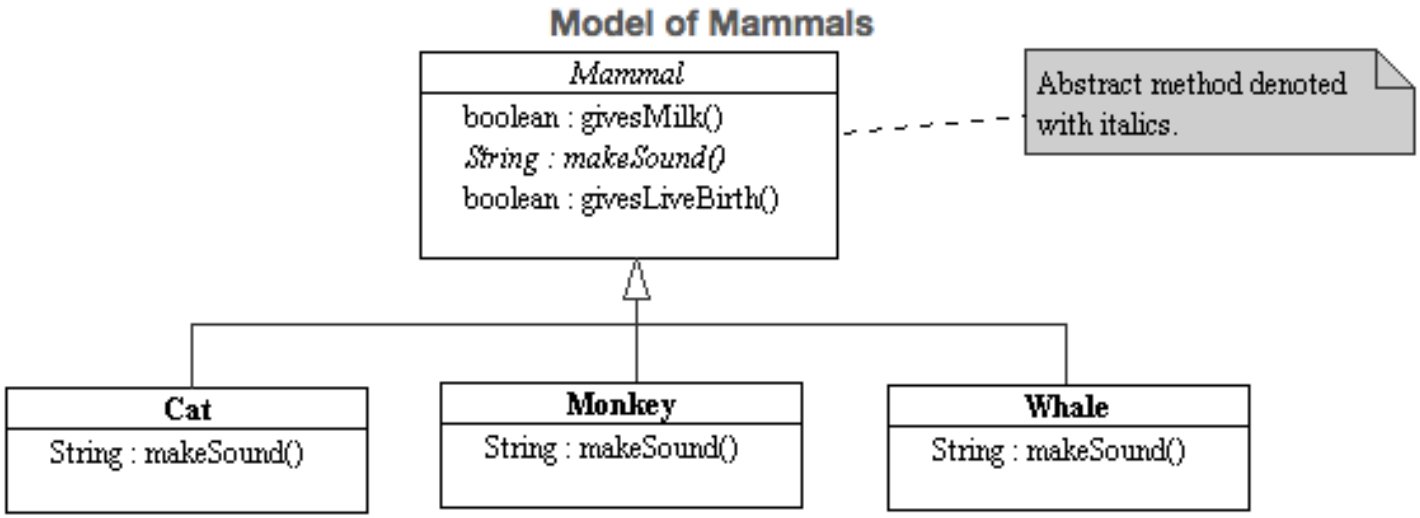
- An abstract class may be defined as the union of all of its concrete subclasses
- Commonality is not the same as abstract equivalence
  - feature may be common to a subset of concrete classes
  - common behaviors should be hoisted to abstract super class



https://cnx.org/contents/YCGmE3x9@11/Union-Design-Pattern-Inheritan

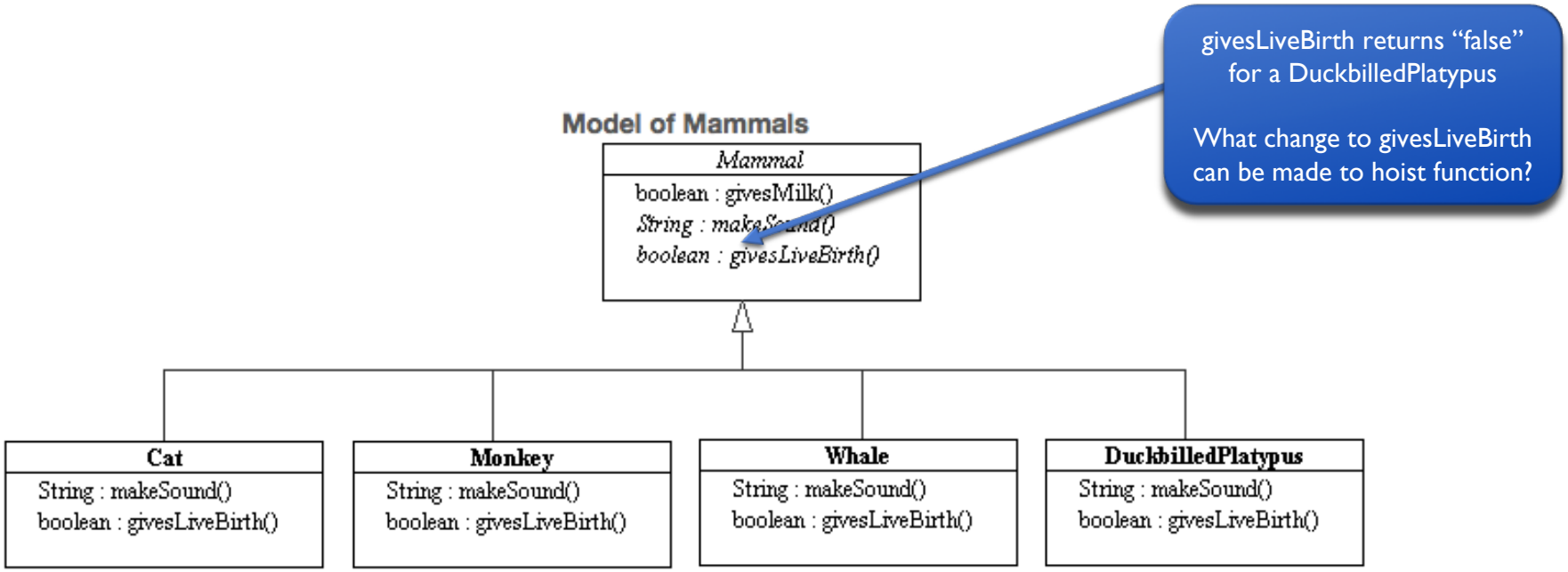
Hoisting Commonality

- Each mammal in subset produces milk and delivers live birth
- Each mammal in subset produces a different sound



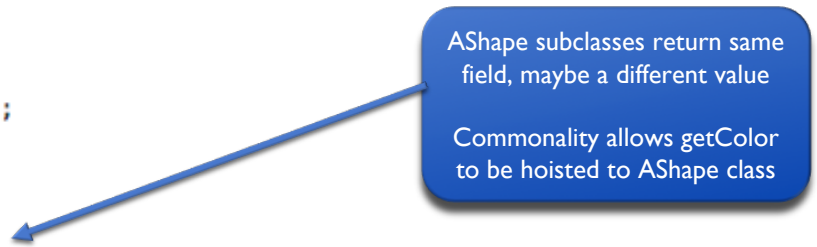
Commonality vs Abstract Equivalence

- DuckbilledPlatypus does not deliver live birth (eggs)
- givesLiveBirth is no longer hoisted to abstract Mammal class
- Each subclass must now implement abstract givesLiveBirth method



AShape Hoisting Example

```
 9  public abstract class AShape {
10      protected Point loc;
11      protected String color;
12
13      /**
14       * Get the shape name
15       * @return shape name
16       */
17      public abstract String getName();
18
19      /**
20       * Get the shape color.
21       * @return shape color
22       */
23      public String getColor() {return this.color; }
24
```



Worksheet #1: MVC Design Review

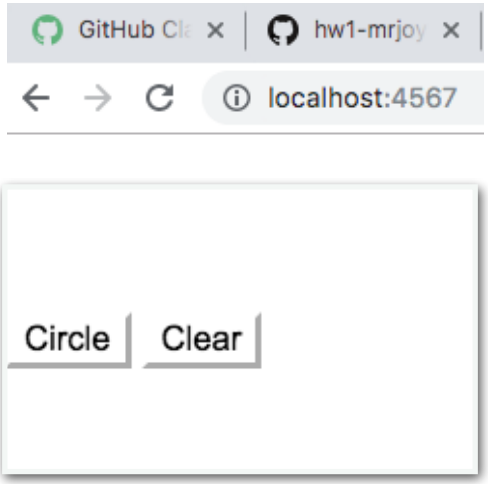
What's wrong with this picture (assume model-view-controller, shapes are defined in the model)

```
9   public abstract class AShape {
10      protected Point loc;
11      protected String color;
12
13      /**
14       * Get the shape name
15       * @return shape name
16       */
17      public abstract String getName();
18
19      /**
20       * Get the shape color.
21       * @return shape color
22       */
23      public String getColor() {return this.color; }
24
25
26      /**
27       * Paint or repaint the shape at a location.  The
28       * lefthand corner of the shape.
29       */
30      public abstract void paint(Point loc, String c);
31  }
```


View Projects with Spark Java and Web Browser

Load index.html for "/" endpoint

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Simple Shapes</title>
6      <script src="js/jquery-3.4.1.min.js"></script>
7      <script src="js/view.js"></script>
8  </head>
9  <body>
10     <div style="position:fixed; top:3em; left:0em;">
11         <button id="btn-circle">Circle</button>
12         <button id="btn-clear">Clear</button>
13         <canvas width="800" height="800"></canvas>
14     </div>
15 </body>
16 </html>
```



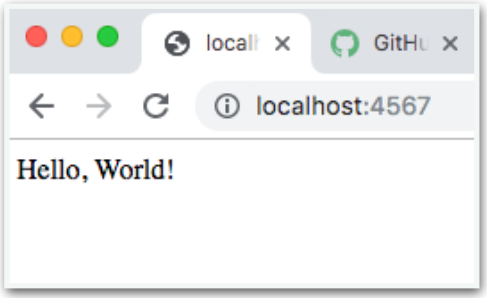
Introduction to Spark Java

- A micro framework for creating web applications (controller)
- Creates a local server at http://localhost:4567
- */hello* endpoint prints "Hello World" on browser using REST get call
- Executes 2 argument *lambda* function

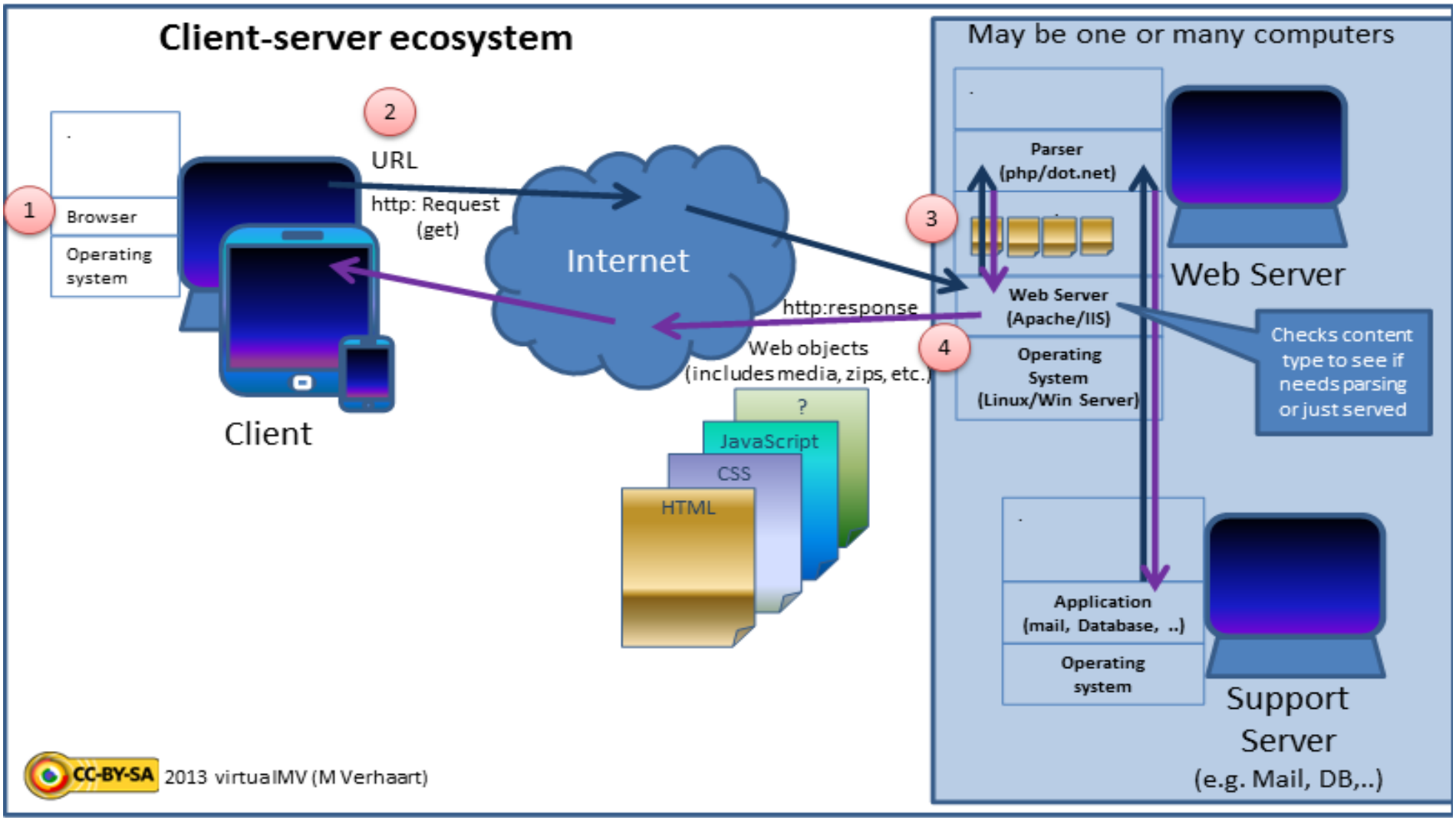
http://sparkjava.com

```
import static spark.Spark.*;

public class HelloWorld {
    public static void main(String[] args) {
        get("/hello", (req, res) -> "Hello World");
    }
}
```



World Wide Web Ecosystem



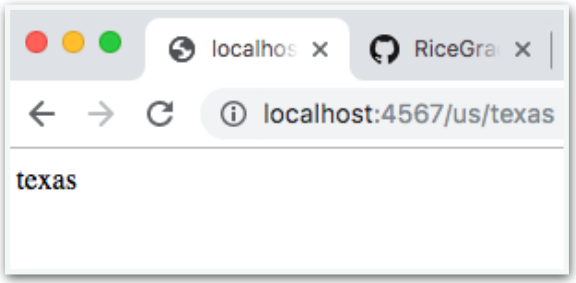
Spark Java REST request options

```
request.attribute("foo");          // value of foo attribute
request.attribute("A", "V");       // sets value of attribute A to V
request.body();                    // request body sent by the client
request.bodyAsBytes();             // request body as bytes
request.contentLength();           // length of request body
request.contentType();             // content type of request.body
request.contextPath();             // the context path, e.g. "/hello"
request.cookies();                 // request cookies sent by the client
request.headers();                 // the HTTP header list
request.headers("BAR");            // value of BAR header
request.host();                    // the host, e.g. "example.com"
request.ip();                      // client IP address
request.params("foo");             // value of foo path parameter
request.params();                  // map with all parameters
request.pathInfo();                // the path info
request.port();                    // the server port
request.protocol();                // the protocol, e.g. HTTP/1.1
request.queryMap();                // the query map
request.queryMap("foo");           // query map for a certain parameter
request.queryParams();             // the query param list
request.queryParams("FOO");        // value of FOO query param
request.queryParamsValues("FOO")   // all values of FOO query param
request.raw();                     // raw request handed in by Jetty
request.requestMethod();           // The HTTP method (GET, ..etc)
request.scheme();                  // "http"
request.servletPath();             // the servlet path, e.g. /result.jsp
request.session();                 // session management
request.splat();                   // splat (*) parameters
request.uri();                     // the uri, e.g. "http://example.com/foo"
request.url();                     // the url. e.g. "http://example.com/foo"
request.userAgent();               // user agent
```


Spark Java REST request useful options

- Visiting a url may trigger request from client to server
- Spark Java server (controller) responds to requests from browser (view)
- Server receives request objects with info describing request
- request.body()
  - get: no body, just a read/retrieval
  - post: may be a body, type is string
- request.params(":state") returns the value associated with ":state"

View



Controller

```
8  public class States {
9
10     /**
11      * Entry point into program.
12      * @param args  The command line arguments.
13      */
14     public static void main(String[] args) {
15         get("/us/:state", (request, response) -> request.params(":state"));
16     }
17 }
```


Grade Policies

- Individual Homework (50%)
- Group Projects (25%)
- Quizzes (10%)
- Design/Communication (10%)
- In-Class Code Exercises (5%)

- 3 slip days can be used for HW1 - HW5. There's no other late policy. Any assignment in which a slip day hasn't been used must be turned in by the deadline.

Announcements & Reminders

- HW 1 will be available on Wednesday, due Friday, Sep 4th at 11:59pm
- Use Piazza (public or private posts, as appropriate) for all COMP 504 communications
- See course web site for syllabus, work assignments, due dates, office hours schedule.