THE TEAM

- Cipta Herwana - Project Manager
- Daniel Lasry - Language Guru
- Nathan Miller - System Architect
- Sam Lee - System Integrator
- Jordan Schau - Tester and Validator
What Is BALL?

- BALL is a language designed to facilitate the simulation of Baseball games for the layman.

- BALL is an imperative and interpreted programming language.

- BALL is built on the efficiencies and powerful toolset of Java.
WHO CAN USE BALL

- Baseball Fans
- Baseball Managers
- Little League Parents
- Sports Bettors
OBJECTIVE

• Create our own custom Simulation functions with ease.

• Streamline the process.

• Omit obfuscating details:
  • Scoping
  • Strong Typing
  • Irrelevant Declarations
CUSTOM SIMULATIONS

• Create a simulation function.
• Activate!
• Call sim(team1, team2, times)

```plaintext
simfunction simpleSim is:
    if (team1’s W > team2’s W) then:
        return team1;
    else:
        return team2;
end
end
activate simpleSim;
sim(Dodgers, Yankees, 5);
```
**Streamlined Process**

- Straightforward loading and management of teams.
- Simple stat declaration and manipulation.
- Easy filtering, sorting and selecting from teams.

```plaintext
stat bWalk = BB / PA;
stat bSingle = (Hits - (2B + 3B + HR)) / PA;
stat bDouble = 2B / PA;
stat bTriple = 3B / PA;
stat bHR = HR / PA;

team Dodgers = load("dodgers.team");

player Manny = "Manny Ramirez" from Dodgers;
print Manny’s bHR;

list 300hitters = Dodgers where (avg > .300);
```
EASE OF PROGRAMMING

• Scope:
  • Functions declared anywhere in the body can be called anywhere!
  • Users do not need to understand scopes.

• Typing:
  • ‘number’ covers integers and decimals.
  • ‘list’, ‘player’ and ‘team’ types.

• Irrelevant Declarations:
  • No need to declare a main class and function.
  • Simple loops and conditionals

```plaintext
function setProb() returns nothing:
    probWalk = 2;
    probSingle = 3;
    probDouble = 4;
end

class number anInteger = 5;
class number aDecimal = 2.34;

class list aList = [team1, team2, team3];
class list aList = [3, 4, 5];

print "play ball";
```
Tools Used

- Byacc/J
- Google Code
- Subversion
- JFlex
- Ubuntu
- Eclipse
Compiler Structure
Compiler Structure
EXAMPLE PROGRAM

simfunction simpleSim is:
    if (team1’s W > team2’s W) then:
        return team1;
    else:
        return team2;
    end
end

activate simpleSim; //activates the simpleSim function

team Indians = load("Indians.team");
team Orioles = load("Orioles.team");

print "Winner: " + sim(Indians, Orioles, 1);
Testing

- Test Driven Development
- White Box Testing
- Regression Testing
- Unit Testing
- Accuracy Testing
- Real Results vs Predicted Results
/*Set the global probabilities of this inning*/
combineProb(batter, pitcher);
randomizeProbabilities();
team1Score += probWalk*(1/4) + probSingle*(1/4) + probDouble*(2/4) + probTriple*(3/4) + probHR;
end

/*NOW, team2 is batting and team1 is pitching! (5 times)*/
do 5 times:
list bestBatters2 = top(9, team2 where (type is "batter"), AVG);
player batter = any bestBatters2;
list bestPitchers2 = bottom(4, team1 where (type is "pitcher"), ERA);
player pitcher = any bestPitchers2;
player batter = any team2 where (type is "batter" and AVG is top 5);
player pitcher = any team1 where (type is "pitcher" and ERA is lowest 5);
combineProb(batter, pitcher);
randomizeProbabilities();
team2Score += probWalk*(1/4) + probSingle*(1/4) + probDouble*(2/4) + probTriple*(3/4) + probHR;
end
if (team1Score > team2Score) then:
print "Team " + team1's name + " wins against team " + team2's name + "!!";
return team1;
else:
print "Team " + team2's name + " wins against team " + team1's name + "!!";
return team2;
end

/*END BASIC SIMULATION FUNCTION*/
Conclusion

• Why BALL?
  • Simple
  • Lightweight
  • Accurate
  • Fun :)

Sunday, December 13, 2009