Programming Assignment 1

CS200 Spring 2012

How to build New Feeds?

- EdgeRank: Algorithm to build a list of news items that are newer and more relevant.

Optimizing the News Feed

Let’s Break Down the Task

<table>
<thead>
<tr>
<th>Assignment #</th>
<th>Objectives</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA2 (Team)</td>
<td>Calculate ER value and sort them</td>
<td>1. Calculating ER score 2. Sorting (QuickSort)</td>
</tr>
<tr>
<td>PA3 (Team)</td>
<td>Extend to multi-user SN</td>
<td>1. Queue of Friends 2. Binary tree of members</td>
</tr>
<tr>
<td>PA4 (Team)</td>
<td>Managing multiple members</td>
<td>1. Build information</td>
</tr>
<tr>
<td>PA5 (Team)</td>
<td>Implementing the News Feed list</td>
<td>1. Calculate ER score for specific person's friend's edge 2. List of news feed</td>
</tr>
</tbody>
</table>

Programming Assignment 1

- Building a Single-person-Social Network Application
- Due date: Feb, 2, 2:00PM
- Late submission: Feb, 3, 2:00PM
- 10% Score deduction
- No submission allowed after Feb. 3, 2:00 pm
- Individual submission
Objective

• Build classes to manage information for a single user.
  – Users name (first name and last name)
  – userID
  – Only the user post’s on their wall
    • For this assignment.
• There is no “friend” yet.
  – Later assignments will expand to let the user’s friends post on the wall.

Tasks

• You will be creating 5 classes
  – Member
  – EdgeStack
  – Edge
  – Item
  – InformationParser
• Modify 1 class
  – PA1

Managing a member

Edges and NewsFeed Objects

• Alice posted a new photo.
  – Item change-status is created by Alice
    • Alice’s wall (homepage) is reference
    • Change-status is item

Edges and NewsFeed Objects

• Bob commented on Alice’s change-status
  – For this Edge
    • Bob’s comment is the item
    • Alice’s change-status is the reference

Member

• Provides methods to retrieve information and update information
  – userID: unique ID for this user
  – First name: user’s first name
  – Last name: user’s last name
  – edgestack: a class maintaining a stack of edges
public class Member {
    private String userID, first, last;
    private EdgeStack edgeStack;
    public Member(){}
    public Member(String _userID, String _first, String _last){}
    public String getUserID(){return userID;}
    public String getFirst(){return first;}
    public String getLast(){return last;}
    public EdgeStack getEdgeStack(){return edgeStack;}
    public void setUserID(String _userID){userID = _userID;}
    public void setFirst(String _first){first = _first;}
    public void setLast(String _last){last = _last;}
    public void setEdgeStack(EdgeStack _edgeStack){edgeStack = _edgeStack;}
}

public class EdgeStack {
    • Stack of Edges
      – Class that maintains a data structure to store and retrieve edges
        • isEmpty()
        • push()
        • pop()
        • peek()
        • create()
        • destroy()
}

public class Edge {
    public Item item;
    public Item reference;
    public Item getItem(){public Item item;public Item reference;public Item getItem();public void setItem(Item item){public void setReference(Item item){public Item item;}public Item item;}public void setItem(Item item){public void setReference(Item item){public Item item;public void setItem(Item item){public void setReference(Item item){public Item item;public void setItem(Item item){public void setReference(Item item){}}}

public class Item {
    • Class maintains information about any object shown on the NewsFeed page. (e.g. posting, comment, picture, etc.)
    • itemID: unique id for this object
    • type: type of the newsfeed object
      1: wall posting 2: comment 3: homepage 4: status change
      5: like 6: visit
    • creator: member ID of the original author
    • message: the content
    • age: age of the item (in second)
public class Item {
    private int itemID;
    private int type;
    private String creator;
    private String message;
    private int age;
    public Item (int itemId, int type, String creator, String message, int age){};
    public int getItemID(){};
    public int getType(){};
    public String getCreator(){};
    public String getMessage(){};
    public int getAge(){};
}

• We use synthesized data to simplify it.
• Data file is provided: PA1-exampleData.txt
• InformationParser should read/parse/create object from this example file.

Format of Example File

member [member id] [first name] [last name]
edge
item [itemID][type][creator][msg][age]
reference [itemID][type][creator][msg][age]
edge
item [itemID][type][creator][msg][age]
reference [itemID][type][creator][msg][age]
edge
item [itemID][type][creator][msg][age]
reference [itemID][type][creator][msg][age]

Parsing Process

• Step 1. Open the file
• Step 2. Read a line
• Step 3. Determine the types of information
• Step 4. Break the line into information
• Step 5. Store the information in an object
• Step 6. Return the object

Useful java classes for parsing

• java.io.File
• java.util.Scanner
• java.lang.String
  – split(): tokenize the string
  – equals(): detect keywords

Example

String[] result = "this is a test".split("\\s");
for (int x=0; x<result.length; x++)
    System.out.println("result["+x+"] =" + result[x]);
result[0] this
result[1] is
result[2] a
result[3] test
PA1.java

- This is the class that includes the main() for this assignment.
- **Do NOT modify the main() method.** We will use this directly for grading.

Main() in PA1.java

```java
public static void main(String args[])
    String input_file = args[0];
    String cmd = args[1];
    PA1 pa1 = new PA1(input_file);
    //test case 1: print name
    if (cmd.equals("print_name")){
        pa1.print_name();
    } else if (cmd.equals("print_userID")){
        //test case 2: print user ID
        pa1.print_userID();
    }else if(cmd.equals("pop_print_EdgeStack")){
        //test case 3: retrieve the nth recently added edge
        //and print current stack
        pa1.pop_print_EdgeStack();
    }else if (cmd.equals("peek_print_Edge")){
        //test case 4: pick nth recently added edge
        //and print that edge
        pa1.peek_print_Edge();
    }else if (cmd.equals("peek_print_EdgeStack")){
        //test case 5: pick nth recently added edge
        //and print current stack
        pa1.peek_print_EdgeStack();
    }
}
```

How to design your software?

- Option 1: Use of the provided skeleton files is optional.
- Option 2: You are encouraged to add more data structure or design patterns (e.g. inheritance)
- Follow the test case scenario. (Do not modify main() in PA1.java)
- Search examples, read them, and try to apply them in your software.
- Ask your TA!

Submitting your PAs

- Use checkin
- **Make a tar ball** with your source code.
  - TA will be looking at your code and design.
- It should compile with following command in the directory you stored your source code
  - e.g. javac *.java
  - Make sure that your source code files are in one flat directory.
- If you did not work on the CS cluster, please make sure that your code works in the **CS linux cluster**.
  - You can ssh to CS machines.