Project Plan

Assignment Hat

Ludachris, Otteran 527604

Shamil Karimli 523001

Inholland University of Applied Sciences

Contents

* 1. Incentive… 1
  2. Objectives 2
  3. Central research question and sub-questions 2
  4. Description of assignment 3
  5. Scope of the assignment including preconditions 5
  6. Deliverables 7
  7. Risks and solutions 8
  8. Stakeholders & Communication Schedule 9
  9. Methods … 10
  10. Planning.. 13
  11. Works Cited 14
  12. Appendices……………………………………………………………………..…………………15

1. **Incentive**

This assignment is about implementing a generic data structure into a code. Analyzing the theoretical complexity of algorithms. Validating the complexity of your algorithms in running time experiments.

The purpose of this assignment is to determine which algorithm suits our case the best.

**2. Objectives**

* implementing a generic data structure;
* analyzing the theoretical complexity of algorithms;
* validating the complexity of your algorithms in running time experiments.

**3. Central Research Question & Sub questions**

## Hat

A hat is a data structure that can be used to retrieve random elements. For instance, it can be used to draw names. It supports the following API:

Public class Hat<Item>

Hat() *create a new empty hat*

boolean isEmpty() *is the hat empty?*

int size() *number of items in the hat*

void add(Item item) *add an item to the hat*

Item draw() *delete a random item from the hat and return it*

#### Example usage:

## To an initially empty hat you add a number of to strings, representing the names of the people in the room. You draw the names of the people who have to do the chores today.

## Hat<String> people = new Hat<String>(); people.add(“Joe”); people.add(“Kate”); people.add(“Mo”); people.add(“Katya”);

## StdOut.println(“Washing dishes: “+people.draw()); StdOut.println(“Vacuuming: “+people.draw());

**4. Description of assignment**

1. Implement the class Hat. Create reasonably efficient implementations of the required methods.
2. Create a simple test client to test your implementations.
3. Determine the average case time complexity of each of the methods. If this is too hard use the worst-case complexity as an upper-bound.
4. Use doubling ratio experiments to verify the time complexity of your implementation.

**5. Scope of the assignment including preconditions**

## Hat

A hat is a data structure that can be used to retrieve random elements. For instance, it can be used to draw names. It supports the following API:

Public class Hat<Item>

Hat() *create a new empty hat*

boolean isEmpty() *is the hat empty?*

int size() *number of items in the hat*

void add(Item item) *add an item to the hat*

Item draw() *delete a random item from the hat and return it*

**6. Deliverables**

|  |  |
| --- | --- |
| Deliverables | Deadline |
| Report | Feb 26 2016  11:59 AM |
| Eclipse project zip | Feb 26 2016  11:59 AM |

**7. Risks and solutions**

Risks:

Running into errors while executing .java file

GitHub Desktop does not commit

Algorithm chosen was extremely inefficient and has to be changed

Solutions:

Check for online recommendation for your platform, or ask one of the group members for help.

If you run into software problems , try writing with another one. If both of them do not work check the root software, and compatibility with your current system.

Chose more efficient algorithm in order to execute the application

**8. Stakeholders and communication schedule**

| **Lecturer** | **Subject** | **Email** | **Deadline** |
| --- | --- | --- | --- |
| **Vera Hollink** | Algorithms & Data structures 1 | [vera.hollink@inholland.nl](mailto:vera.hollink@inholland.nl) | Feb 26 2016  11:59 AM |

**9. Methods**

Methods described in order to finish the assessment

**10. Planning**

Deadline : Friday Feb 26 11:59 CET

**11. Works Cited**

Assignment Description on blackboard by Vera Hollink

Literature:

Cay S.Horstmann ,Gary Cornell, Core Java Volume I Fundamentals Ninth Edition

Robert Sedgewick and Kevin Wayne . Algorithms Fourth Edition

**12. Appendices**

hat.java

import java.util.Random;

public class Hat<T> {

private T[] elements;

private int N;

private final int size = 4;

private Object obj;

public Hat() {

elements = (T[]) new Object[size];

}

public void add(T t) {

elements[N++] = t;

}

@SuppressWarnings("unchecked")

public T draw() {

obj = elements[new Random().nextInt(elements.length)];

for(int i = 0; i < N; i++){

if(obj.equals(elements[i])){

elements[i] = elements[--N];

}

}

return (T) obj;

}

public boolean isEmpty(){

return N == 0;

}

public int size(){

return N;

}

}

hat\_test\_client.java

public class hat\_test\_client {

public static void main(String[] args) {

Hat<String> people = new Hat<String>();

people.add("Ama");

people.add("Kofi");

people.add("Yaa");

people.add("Adwoa");

System.out.println("Washing dishes: " + people.draw());

System.out.println("Vacuuming: " + people.draw());

System.out.println(people.size());

}

}