Group AA Project Two Proposal

GROUP: AA	TA NAME: Rachit
FLIPGRID: https://flip.com/723b78ec	TA EMAIL: rtibrewal@wisc.edu

Name	Email	Team	P1 Role	P2 Role	P3 Role
Rachit Bandaram	rbandaram@wisc.edu	Red	BD	DW	FD
Nakul Balaji	nbalaji@wisc.edu	Red	FD	AE	BD
Rohan Babaria	rbabaria@wisc.edu	Red	FD	BD	AE
Noah Lessard	nclessard@wisc.edu	Red	BD	FD	DW
Dhruv Arora	darora25@wisc.edu	Blue	BD	DW	FD
Sebio Armendariz-kerr	earmendarizk@wisc.edu	Blue	FD	AE	BD
Masa Abboud	mabboud@wisc.edu	Blue	FD	BD	AE
Cameron Abplanalp	cabplanalp@wisc.edu	Blue	BD	FD	DW

Project Title: Generic Card Trader

Brief Project Description:

Our project is an application that stores trading card data, which consists of a unique identifier and a monetary value. This project will utilize a red-black tree effectively because a feature will be getting and iterating cards based on the value of cards from highest to lowest, while still being able to continuously insert and remove values and maintain a high efficiency, a task that can be implemented with a red-black binary search tree. This ensures a greater level of efficiency than a sorted array, because an array would not be as efficient when inserting/removing new cards, after the initial data has been loaded. This application will support insertion by loading a file or adding in a frontend interface, inserting/removing single cards, high/low statistics, and printing a summary of your stored deck .

Representative Tasks Performed Using this Application:

1. Loading Data from a file (demo by DW)
2. Accurate single insertion/~~removal~~ after initial data has been loaded (demo by AE)
3. Accurate iteration through single inserted cards from specific value (demo by BD)
4. Loading single data, printing accurate summary (demo by FD)

Data Wrangler (DW) Role: **Rachit Bandaram, Dhruv Arora**

- Create/Find/Handle a large CSV file full of different card games (ID's / Stats) tables
- Convert from CSV to an array of BD generic class card objects (`TradingCard`)
- Each line in the CSV file will contain all the information needed to create Card Trader object (`TradingCard`)
- We will use a delimiter (,) to separate the different data fields

Data Description:

The data being loaded into this project will be a CSV file of a generic type of card game. For this proposal, a CSV file will be created from pokemon card data scraped from online. The format will be <Name,MonetaryValue> on each line of the CSV file, where Name is the name of the pokemon / trading card and MonetaryValue is the cost of the pokemon / trading card found here. The DW Object will be able to parse this data from a string filename and turn it into a list of `TradingCard` generic objects for the rest of the project to use.

Development Responsibilities (Interface):
```
import java.io.FileNotFoundException;
import java.util.List;
public interface DWInterface {

    // public CardDW();
    public List<TradingCard> readValuesFromFile(String filename)
throws FileNotFoundException; {}

}
```

Presentation Responsibilities:

- Create a video showing that this role's JUnit5 tests are running, and also show that data can be loading from a file into an array of the generic class TradingCard

---

Algorithm Engineer (AE) Role: **Nakul Balaji, Sebio Armendariz-kerr**

- Use BD generic class (`TradingCard`)
- Add size/total value fields (update in insert)
- Add getters for size and total value
- Updating remove functions to rebalance tree and maintain efficiency (remember to change total value of "deck" when you call)
- Review getHighest / getLowest
- Adding the interable functions
- Update inser

Capabilities Added to Required Data Structure:

The Algorithm Engineer will extend the provided SortedCollectionInterface to add additional functionality, and also use the `TradingCard` generic object. The RB-Tree will have efficient removal/insertion, and also be able to iterate through the RB tree based on a given start value. This iteration function will be given a starting node, and the progress through the right of the tree until the end, then return all those nodes as a list of trading cards. This will be used in the application's ability to get all cards above a certain value.

Development Responsibilities (Interface):

```
public interface AEInterface implements
SortedCollectionInterface()<TradingCard>{

// public CardAE();

public int size; //already defined in collection
private double totalValue; //update in insert/remove

public double getSize();
public double getTotalValue();

public TradingCard getHigest();
public TradingCard getLowest();

public List<TradingCard> iterateOver (double startValue) {};

// already defined in collection
Public boolean isEmpty();
Public boolean contains(TradingCard data); {}
public boolean insert(TradingCard data) throws NullPointerException,
IllegalArgumentException {};
public boolean remove(TradingCard data) throws NullPointerException,
IllegalArgumentException {};
}
```

Presentation Responsibilities:

- Create a video showing that this role's JUnit5 tests are running, and also show that generic TradingCard objects can be inserted/~~removed~~ after initial data has been loaded in the RB-DST.

Backend Developer (BD) Role: **Rohan Babaria, Masa Abboud**

- Connect AE functionality (inserting/removing array of generic objects into RB-BST)
- Create generic trading card class (`TradingCard`) that other roles will use
- Call get Highest/Lowest and return formatted string
- Call methods with mean / total range value, return formatted string
- Create a sorting method with an iterable and return array of generic objects.

Backend Functionality Description:

The backendDeveloper is responsible for creation of a backend interface that can get various different statistics, pass in a file pathway for the data wrangler to load, and use the AE interface to print out a collection of trading cards above a certain value through the use of their iterable class that starts at a given minimum node. Also, The backend is responsible for creating a TradingCard Object class that stores data about a given TradingCard, including its value and its ID. This object implements comparable and is used as our objects stored in the algorithm engineers tree.

Development Responsibilities (Interface):

```
public interface BDInterface{
// public CardBackend(CardDataWrangler data, CardAE tree,);
/*
@throws IllegalArgumentException
*/
public TradingCard[] sortCards(double minVal);
public void loadData(String data) throws FileNotFoundException;
public TradingCard getLow();
public TradingCard getHigh();
public double getRange();
public double getValue();
public double getMean();
public TradingCard remove(t id, double value) throws IllegalArgument
Exception,NoSuchElementException;
public void insert(t id, double value)IllegalArgumentException;
}T
public interface TradingCardInterface<T> extends Comparable<Object> {
  // value of a trading card must be non-null and positive
  // public TradingCard(T ID, double value) throws
IllegalArgumentException
  /**
   * Returns the monetary value of the trading card
   *
   * @return the value of a trading card
   */
  public double getValue();
  /**
   *
   * @return the ID of the TradingCard
   */
  public T getID();
  /**
   *
   * @return returns the card to a string formatted in (ID, Value)
   */
  public String toString();
  /**
   * checks if one card is the same as the other (same id and value)
   *
   * @param o trading card to compare
   * @return true if the cards are equal and false otherwise
   * @throws IllegalArgumentException if the object is not a
TradingCard object
   */
  public boolean Equals(Object o) throws IllegalArgumentException;
  /**
   * Compares your TradingCard to OtherTradingCard. if
OtherTradingCard is greater than your card,
   * return -1 if OtherTradingCard is less than your card, return 1
if OtherTradingCard and your
   * card are equal, return 0
   *
   *
   * @throws IllegalArgumentException if the object is not a
TradingCard object
   */
  public int compareTo(Object OtherTradingCard) throws
IllegalArgumentException;
}
```

Presentation Responsibilities:

- Create a video showing that this role's JUnit5 tests are running, and also show that iterating over the RB-BST is accurate after single inserting/~~removing~~.

Frontend Developer (FD) Role: **Noah Lessard, Cameron Abplanalp**

- Create an interactive main menu containing following commands:
  - Get summary (highest, lowest, mean, total range)
  - Get all cards above a certain value (exclusive)
  - Insert Cards from File
  - Insert Cards from collection
  - Remove Card from collection
  - Quit

Log of a Sample Execution of the App:

```
Welcome to the Generic Card Sorting App
      [L]oad a card collection from a CSV file
      [I]nsert a single card into the current collection
      [R]emove a single card and return value of removed card
      [P]rint a summary of statistic of the current collection
      [G]et all cards above a certain value
      [Q]uit the program
```

Development Responsibilities (Interface):

```
public interface FDInterface {
// public CardFrontend(Scanner input, BDInterface backend);
  public void hr();
  public void commandLoop();
  public String printPromptingMenu();
  public void loadDataFromFile(String filename);
  public void insertSingleData(Object ID, double value);
  public TradingCardInterface removeSingleData(Object ID, double
value);
  public void displaySummary();
  public TradingCardInterface[] getCardsAboveValue(double value);
}
```

Presentation Responsibilities:

- Create a video showing that this role’s JUnit5 tests are running, and also show loading single TradingCard objects, and printing an accurate summary, with no exceptions being thrown from invalid inputs.

Scope and Signatures:

Ideas for Scoping Up:

- Adding more stats to each card, wear and tear, distinguish between card games?
- Multiple RB-BST per application, one for each type of card (pokemon, magic, baseball).

Ideas for Scoping Down:

- Focus more on more specific statistics, less general.
- No removal function, just focus on iteration, or vice versa.

Outside Libraries and Other Tools:

No libraries outside of the standard java library will be used. We will use this dataset.

Team Signatures:

Name	Email	Team	Type Name As Signature
Rachit Bandaram	rbandaram@wisc.edu	Red	
Nakul Balaji	nbalaji@wisc.edu	Red	Nakul Balaji
Rohan Babaria	rbabaria@wisc.edu	Red	Rohan Babaria

Noah Lessard	nclessard@wisc.edu	Red	Noah Lessard
Dhruv Arora	darora25@wisc.edu	Blue	Dhruv Arora
Sebio Armendariz-kerr	earmendarizk@wisc.edu	Blue	
Masa Abboud	mabboud@wisc.edu	Blue	Masa Abboud
Cameron Abplanalp	cabplanalp@wisc.edu	Blue	Cameron Abplanalp

TA Feedback:

The project description and demonstration should use the insert/remove function of RBTree after the initial data is loaded from file. This is essential as otherwise one could make use of a sorted array instead of an RBTree.

Team Response:

We clarified language in the project description and demonstration to better represent how the insert/remove function of the RBTree will be used to increase efficiency over a sorted array.

Proposal Amendments:

If your group needs to make any changes to what is described above after the proposal deadline, then 1) make sure everyone in your group agrees with those changes, 2) describe those changes in the first empty row below, and then 3) notify your group's TA about those changes and whey are being made. Your TA will then review your request and indicate whether they approve of such changes by adding their initials to the end of that amendment's row below.

Number	Description	TA Approval
1	Get rid of remove() and remove functionality	
2		
3		
4		

5		
6		
7		
8		