UNIVERSITY AT ALBANY
State University of New York

COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE
CSI213 Data Structures

Project 04 Created by Qi Wang

Table of Contents

Part I: General Project Information

- All projects are individual projects unless it is notified otherwise.
- All projects must be submitted via Blackboard. No late projects or e-mail submissions or hard copies will be accepted.
- Unlimited submission attempts will be allowed on Blackboard. Only the last attempt will be graded.
- Work will be rejected with no credit if
    - The project is late.
    - The project is not submitted properly (wrong files, not in required format, etc.). For example,
        - The submitted file can't be opened.
        - The submitted work is empty or wrong work.
        - Other issues.
    - The project is a copy or partial copy of others' work (such as work from another person or the Internet).
- Students must turn in their original work. Any cheating violation will be reported to the college. Students can help others by sharing ideas, but not by allowing others to copy their work.
- Documents to be submitted as a zipped file:
    - **UML class diagram(s)** – created with Violet UML or StarUML
    - **Java source file(s) with Javadoc style inline comments**
    - **Supporting files if any** (For example, files containing all testing data.)
- Students are required to submit a design, all error-free source files with Javadoc style inline comments, and supporting files. Lack of any of the required items will result in a really low credit or no credit.
- Your TA will grade and post the feedback and the grade on Blackboard if you have submitted the work properly and on time. If you have any questions regarding the feedback or the grade, please reach out to the TA first. You may also contact the instructor for this matter.

Part II: Project grading rubric

This project is selected for assessing the following student outcome:

SO2: Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

SO2	**Levels of Performance**			
	UNSATISFACTORY	**DEVELOPING**	**SATISFACTORY**	**EXEMPLARY**
Performance Indicator #1: **The software meets the specifications.** The design is efficient. Specifications and implementation are separated. All classes are completed correctly. All required functionalities are completed correctly. All source codes are well formatted and documented with correct Javadoc style comments.	The software doesn't compile and/or doesn't include all requirements.	The software compiles but it doesn't meet all requirements or there are issues.	The software compiles, meets all the requirements with some minor issues.	The software compiles, meets all the requirements with no issues.
	/5	/14	/24	/35
Performance Indicator #2: The design is efficient, and the UML diagram is correct. For all class diagrams, visibility, name, return type/parameter type are correct for each member of a class. Class relationships are correct.	The design is efficient, and the UML diagram is incorrect.	The design is efficient, and the UML diagram is correct for some classes.	The design is efficient, and the UML diagram is correct with some issues.	The design is efficient, and the UML diagram is correct with no issues.
	/2	/4	/7	/10
Performance Indicator #3: The software is completely tested.	The implementation is not completed or tested.	The implementation is somewhat completed and some classes are tested.	The implementation is completed and all classes are tested with minor issues.	The implementation is completed and all classes are tested with no issues.
	/2	/6	/10	/15
Level of Performance Column Total				

Total Points Awarded	
Feedback to the student by TA/Grader:	**Analysis:** • Does the software meet the exact specification / customer requirements? • Does the software solve the exact problem? **Design:** • Is the design efficient? **Code:** • Are there errors? • Are code conventions followed? • Does the software use the minimum computer resource (computer memory and processing time)? • Is the software reusable? • Are comments completely written in Javadoc format? a. Class Javadoc comments must be included before a class header. b. Method Javadoc comments must be included before a method header. c. More inline comments (in either single line format or block format) must be included inside each method body. d. All comments must be completed in correct format such as tags, indentation etc. **Debug/Testing:** • Are there bugs in the software? **Documentation:** • Complete all documentations that are required. **<u>Overall Feedback:</u>**

Part III: Examples of UML diagrams and Javadoc style comments

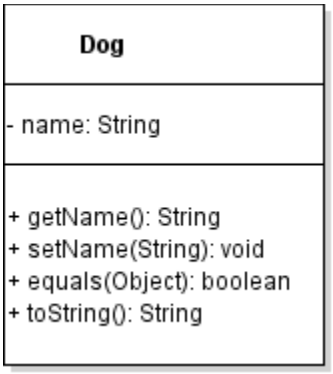
To complete a project, the following steps of a software development cycle should be followed. These steps are not pure linear but overlapped.

Analysis-design-code-test/debug-documentation.

1) Read project description to understand all specifications(**Analysis**).
2) Create a design (an algorithm for method or a UML class diagram for a class) (**Design**)
3) Create Java programs that are translations of the design. (**Code/Implementation**)
4) Test and debug, and (**test/debug)**
5) Complete all required documentation. (**Documentation**)

The following shows a sample design. By convention.

- *Constructors* and *constants* should not be included in a class diagram.
- For each *field* (instance variable), include *visibility*, *name*, and *type* in the design.
- For each *method*, include *visibility*, *name*, *parameter type(s)* and *return type* in the design.
    - DON'T include *parameter names*, only *parameter types* are needed.
- Show class relationships such as *dependency*, *inheritance*, *aggregation*, etc. in the design. Don't include the *driver* program and its helper class since they are for testing purpose only.



```
import java.util.Random;

/**
 * Representing a dog with a name.
 * @author Qi Wang
 * @version 1.0
 */
public class Dog{
	/**
	 * The name of this dog
	 */
	private String name;


	/**
	 * Constructs a newly created Dog object that represents a dog with an empty name.
	 */
	public Dog(){
		this("");
	}
	/**
	 * Constructs a newly created Dog object wit
	 * @param name The name of this dog
	 */
	public Dog(String name){
		this.name = name;
	}

	/**
	 * Returns the name of this dog.
	 * @return The name of this dog
	 */
	public String getName(){
		return this.name;
	}
	/**
	 * Changes the name of this dog.
	 * @param name The name of this dog
	 */
	public void setName(String name){
		this.name = name;
	}
	/**
	 * Returns a string representation of this dog. The returned string contains the type of
	 * this dog and the name of this dog.
	 * @return A string representation of this dog
	 */
	public String toString(){
		return this.getClass().getSimpleName() + ": " + this.name;
	}
	/**
	 * Indicates if this dog is "equal to" some other object. If the other object is a dog,
	 * this dog is equal to the other dog if they have the same names. If the other object is
	 * not a dog, this dog is not equal to the other object.
	 * @param obj A reference to some other object
	 * @return A boolean value specifying if this dog is equal to some other object
	 */
	public boolean equals(Object obj){
		//The specific object isn't a dog.
		if(!(obj instanceof Dog)){
			return false;
		}

		//The specific object is a dog.
		Dog other = (Dog)obj;
		return this.name.equalsIgnoreCase(other.name);
	}
}
```

Class comments must be written in Javadoc format before the class header. A **description** of the class, author information and version information are required.

Comments for fields are required.

open {

TAB

TAB

Method comments must be written in Javadoc format before the method header. the first word must be a **capitalized** verb in the **third** person. Use punctuation marks properly.

open {

TAB

TAB

A description of the method, comments on parameters if any, and comments on the return type if any are required.

A Javadoc comment for a formal parameter consists of three parts:

- parameter tag,
- a name of the formal parameter in the design , (The name must be consistent in the comments and the header.)
- and a phrase explaining what this parameter specifies.

A Javadoc comment for return type consists of two parts:

- return tag,
- and a phrase explaining what this returned value specifies

More inline comments can be included in single line or block comments format in a method.

Part IV Project description

In this project, you will design, implement and test two ADTs: ADT Binary Search Tree and ADT Address book. Please read the following requirements.

A. <u>ADT Binary Search tree:</u>

A binary search tree can be used to store a list of **comparable** objects. In a generic binary search tree class, constrain E(the type parameter) to be the class types that implement *Comparable* interface. In the class header, add the interface as the upper bound to E.

```
public class BinarySearchTree<E extends Comparable<E>>
```

Specifications of the ADT binary search tree:

- Construct a binary search tree.
- **Retrieve the root of this tree.**
- **Change the root of this tree if supported.**
- **Check if this tree is empty.**
- **Make this tree empty.**
- Search an element in this tree.
- Insert an element into this tree.
- Delete an element from this tree.
- Construct an iterator of this tree.
- Get a reference to the element of a node.

You should design these operations in bold in a super class and put the rest in a sub class. To support this class, the following classes must be designed and implemented:

- Generic tree node class: a tree node contains an element, a left reference, and a right reference.
- Generic tree iterator class: a class implementing *Iterator* interface and provides traversal operations.
- Tree exception class: used for the abnormal situations from some tree operations.

These classes must be tested first before they will be used in the second ADT: ADT address book. For testing, start with an empty binary search tree, insert a list of integers, invoke other methods to test completely. Write a driver and a helper class for the driver.

```
public static void start()…{
      BinarySearchTree<Integer> tree = new BinarySearchTree<Integer>();
      //Fill the tree.
      create(list);
      //Display the tree.
      display(list);
      …
}
```

B. **<u>ADT Address Book</u>:**

An ADT Address book maintains a list of contacts (friends and families) in a binary search tree. The ADT is not a generic class since the element of a node is a contact. This ADT is implemented using a binary search tree as the data

structure. You will use the binary search tree from part A. The following shows values a contact contains. Assume that all contacts have unique full names.

first_name	last_name	street	city	state	zipcode	phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927

Specifications of the ADT address book:

Note: most of the operations are executed in the underlying binary search tree of this address book.

- Construct an address book.
- Search a contact in this address book.
- Insert a contact into this address book.
- Delete a contact from this address book.
- Check if this address book is empty.
- Make this address book empty.
- Find all contacts in the same zip code and return them sorted in a linked list(use java.util.LinkedList<E>). A specific zip code is passed into the method as a parameter.
- Find all contacts in the same city and return them sorted in a linked list(use java.util.LinkedList<E>). A specific city is passed into the method as a parameter.
- Find all contacts whose phone numbers have the same area code and return them sorted in a linked list(use java.util.LinkedList<E>). A specific area code is passed into the method as a parameter.

To support the ADT Address book, two more classes are required: *Address* and *Contact*.

- An address contains *street*, *city*, *state* and *zip code*. Write a class to represent an address object.

first_name	last_name	street	city	state	zipcode	phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927

- A contact contains *a full name*, *an address(*an *Address* object*)* and *a phone number*. Write a class to represent a contact object.

first_name	last_name	address				phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927

Now it is time to test the ADT address book. For testing, Write a driver and a helper class for the driver. Start with an empty address book, fill the address book with contact objects.

```
public static void start()…{
            AddressBook list = new AddressBook();
            //Fill the address book.
            create(list);
            //Display the address book.
            display(list);
            …
}
```

Use the contact input data on the next page to make contact objects for testing. Please copy the data into a text file first.

first_name	last_name	street	city	state	zipcode	phone
James	Butt	6649 N Blue Gum St	New Orleans	LA	70116	504-621-8927
Josephine	Darakjy	4 B Blue Ridge Blvd	Brighton	MI	48116	810-292-9388
Art	Venere	8 W Cerritos Ave #54	Bridgeport	NJ	8014	856-636-8749
Lenna	Paprocki	639 Main St	Anchorage	AK	99501	907-385-4412
Donette	Foller	34 Center St	Hamilton	OH	45011	513-570-1893
Simona	Morasca	3 Mcauley Dr	Ashland	OH	44805	419-503-2484
Mitsue	Tollner	7 Eads St	Chicago	IL	60632	773-573-6914
Leota	Dilliard	7 W Jackson Blvd	San Jose	CA	95111	408-752-3500
Sage	Wieser	5 Boston Ave #88	Sioux Falls	SD	57105	605-414-2147
Kris	Marrier	228 Runamuck Pl #2808	Baltimore	MD	21224	410-655-8723
Minna	Amigon	2371 Jerrold Ave	Kulpsville	PA	19443	215-874-1229
Abel	Maclead	37275 St Rt 17m M	Middle Island	NY	11953	631-335-3414
Kiley	Caldarera	25 E 75th St #69	Los Angeles	CA	90034	310-498-5651
Graciela	Ruta	98 Connecticut Ave Nw	Chagrin Falls	OH	44023	440-780-8425
Cammy	Albares	56 E Morehead St	Laredo	TX	78045	956-537-6195
Mattie	Poquette	73 State Road 434 E	Phoenix	AZ	85013	602-277-4385
Meaghan	Garufi	69734 E Carrillo St	Mc Minnville	TN	37110	931-313-9635
Gladys	Rim	322 New Horizon Blvd	Milwaukee	WI	53207	414-661-9598
Yuki	Whobrey	1 State Route 27	Taylor	MI	48180	313-288-7937
Fletcher	Flosi	394 Manchester Blvd	Rockford	IL	61109	815-828-2147
Bette	Nicka	6 S 33rd St	Aston	PA	19014	610-545-3615
Veronika	Inouye	6 Greenleaf Ave	San Jose	CA	95111	408-540-1785
Willard	Kolmetz	618 W Yakima Ave	Irving	TX	75062	972-303-9197
Maryann	Royster	74 S Westgate St	Albany	NY	12204	518-966-7987
Alisha	Slusarski	3273 State St	Middlesex	NJ	8846	732-658-3154
Allene	Iturbide	1 Central Ave	Stevens Point	WI	54481	715-662-6764
Chanel	Caudy	86 Nw 66th St #8673	Shawnee	KS	66218	913-388-2079
Ezekiel	Chui	2 Cedar Ave #84	Easton	MD	21601	410-669-1642
Willow	Kusko	90991 Thorburn Ave	New York	NY	10011	212-582-4976
Bernardo	Figeroa	386 9th Ave N	Conroe	TX	77301	936-336-3951
Ammie	Corrio	74874 Atlantic Ave	Columbus	OH	43215	614-801-9788
Francine	Vocelka	366 South Dr	Las Cruces	NM	88011	505-977-3911
Ernie	Stenseth	45 E Liberty St	Ridgefield Park	NJ	7660	201-709-6245
Albina	Glick	4 Ralph Ct	Dunellen	NJ	8812	732-924-7882
Alishia	Sergi	2742 Distribution Way	New York	NY	10025	212-860-1579
Solange	Shinko	426 Wolf St	Metairie	LA	70002	504-979-9175
Jose	Stockham	128 Bransten Rd	New York	NY	10011	212-675-8570
Rozella	Ostrosky	17 Morena Blvd	Camarillo	CA	93012	805-832-6163
Valentine	Gillian	775 W 17th St	San Antonio	TX	78204	210-812-9597
Kati	Rulapaugh	6980 Dorsett Rd	Abilene	KS	67410	785-463-7829
Youlanda	Schemmer	2881 Lewis Rd	Prineville	OR	97754	541-548-8197
Dyan	Oldroyd	7219 Woodfield Rd	Overland Park	KS	66204	913-413-4604
Roxane	Campain	1048 Main St	Fairbanks	AK	99708	907-231-4722
Lavera	Perin	678 3rd Ave	Miami	FL	33196	305-606-7291

Erick	Ferencz	20 S Babcock St	Fairbanks	AK	99712	907-741-1044
Fatima	Saylors	2 Lighthouse Ave	Hopkins	MN	55343	952-768-2416
Jina	Briddick	38938 Park Blvd	Boston	MA	2128	617-399-5124
Kanisha	Waycott	5 Tomahawk Dr	Los Angeles	CA	90006	323-453-2780
Emerson	Bowley	762 S Main St	Madison	WI	53711	608-336-7444
Blair	Malet	209 Decker Dr	Philadelphia	PA	19132	215-907-9111
Brock	Bolognia	4486 W O St #1	New York	NY	10003	212-402-9216
Lorrie	Nestle	39 S 7th St	Tullahoma	TN	37388	931-875-6644
Sabra	Uyetake	98839 Hawthorne Blvd #6101	Columbia	SC	29201	803-925-5213
Marjory	Mastella	71 San Mateo Ave	Wayne	PA	19087	610-814-5533
Karl	Klonowski	76 Brooks St #9	Flemington	NJ	8822	908-877-6135
Tonette	Wenner	4545 Courthouse Rd	Westbury	NY	11590	516-968-6051
Amber	Monarrez	14288 Foster Ave #4121	Jenkintown	PA	19046	215-934-8655
Shenika	Seewald	4 Otis St	Van Nuys	CA	91405	818-423-4007
Delmy	Ahle	65895 S 16th St	Providence	RI	2909	401-458-2547
Deeanna	Juhas	14302 Pennsylvania Ave	Huntingdon Valley	PA	19006	215-211-9589
Blondell	Pugh	201 Hawk Ct	Providence	RI	2904	401-960-8259
Jamal	Vanausdal	53075 Sw 152nd Ter #615	Monroe Township	NJ	8831	732-234-1546
Cecily	Hollack	59 N Groesbeck Hwy	Austin	TX	78731	512-486-3817
Carmelina	Lindall	2664 Lewis Rd	Littleton	CO	80126	303-724-7371
Maurine	Yglesias	59 Shady Ln #53	Milwaukee	WI	53214	414-748-1374
Tawna	Buvens	3305 Nabell Ave #679	New York	NY	10009	212-674-9610
Penney	Weight	18 Fountain St	Anchorage	AK	99515	907-797-9628
Elly	Morocco	7 W 32nd St	Erie	PA	16502	814-393-5571
Ilene	Eroman	2853 S Central Expy	Glen Burnie	MD	21061	410-914-9018
Vallie	Mondella	74 W College St	Boise	ID	83707	208-862-5339
Kallie	Blackwood	701 S Harrison Rd	San Francisco	CA	94104	415-315-2761
Johnetta	Abdallah	1088 Pinehurst St	Chapel Hill	NC	27514	919-225-9345
Bobbye	Rhym	30 W 80th St #1995	San Carlos	CA	94070	650-528-5783
Micaela	Rhymes	20932 Hedley St	Concord	CA	94520	925-647-3298
Tamar	Hoogland	2737 Pistorio Rd #9230	London	OH	43140	740-343-8575
Moon	Parlato	74989 Brandon St	Wellsville	NY	14895	585-866-8313
Laurel	Reitler	6 Kains Ave	Baltimore	MD	21215	410-520-4832
Delisa	Crupi	47565 W Grand Ave	Newark	NJ	7105	973-354-2040
Viva	Toelkes	4284 Dorigo Ln	Chicago	IL	60647	773-446-5569
Elza	Lipke	6794 Lake Dr E	Newark	NJ	7104	973-927-3447
Devorah	Chickering	31 Douglas Blvd #950	Clovis	NM	88101	505-975-8559
Timothy	Mulqueen	44 W 4th St	Staten Island	NY	10309	718-332-6527
Arlette	Honeywell	11279 Loytan St	Jacksonville	FL	32254	904-775-4480
Dominque	Dickerson	69 Marquette Ave	Hayward	CA	94545	510-993-3758
Lettie	Isenhower	70 W Main St	Beachwood	OH	44122	216-657-7668
Myra	Munns	461 Prospect Pl #316	Euless	TX	76040	817-914-7518
Stephaine	Barfield	47154 Whipple Ave Nw	Gardena	CA	90247	310-774-7643
Lai	Gato	37 Alabama Ave	Evanston	IL	60201	847-728-7286
Stephen	Emigh	3777 E Richmond St #900	Akron	OH	44302	330-537-5358
Tyra	Shields	3 Fort Worth Ave	Philadelphia	PA	19106	215-255-1641

Tammara	Wardrip	4800 Black Horse Pike	Burlingame	CA	94010	650-803-1936
Cory	Gibes	83649 W Belmont Ave	San Gabriel	CA	91776	626-572-1096
Danica	Bruschke	840 15th Ave	Waco	TX	76708	254-782-8569
Wilda	Giguere	1747 Calle Amanecer #2	Anchorage	AK	99501	907-870-5536
Elvera	Benimadh	99385 Charity St #840	San Jose	CA	95110	408-703-8505
Carma	Vanheusen	68556 Central Hwy	San Leandro	CA	94577	510-503-7169
Malinda	Hochard	55 Riverside Ave	Indianapolis	IN	46202	317-722-5066
Natalie	Fern	7140 University Ave	Rock Springs	WY	82901	307-704-8713
Lisha	Centini	64 5th Ave #1153	Mc Lean	VA	22102	703-235-3937