CSE 12 Note #1 Introduction, Generics, ADT, Junit

Important course sites

1. canvas.ucsd.edu is our material repo and gradebook.
2. gradescope.com is our homework submission site
3. piazza.com is our discussion forum
4. autograder.ucsd.edu is our tutor help site (also has course calendar)
5. zybooks is our textbook and reading site (has to link from canvas to zybooks!)
6. Course syllabus: https://bit.ly/12cao-syllabus
7. Course schedule: https://bit.ly/12cao-schedule

Import course logistics (refer to syllabus)

Instructor: Paul Cao
Contact: EBU 3B 2102, yic242@ucsd.edu [https://sites.google.com/ucsd.edu/paul-cao/]
Office hours: Friday 1pm - 3pm (in person in CSE 2102)

Topics

- Java Collections Library
- The use of generics, interfaces. abstraction, ADTs, invariants
- OO programming idioms: input and output parameters
- Exception design and handling
- Runtime analysis
- Use of dictionary and set
- Use of binary search trees
- Implementation of basic trees using recursion
- Analysis of comparison-based sorting using the following data structures
    - Heaps
    - Arrays
    - Trees (binary trees)

Grading

- Reading assignments from zyBooks (drop 1 week): 10%
- Class participation (drop 3 participations and start to count participation in week 2): 5%
- Quizzes: 10%
- Midterm: 10%
- PA: 40%
- Final exam (a portion of the final can be used to sub the midterm): 25%

Academic Integrity

- All work individually done
- No public github repo pls
- Penalty for AI violation
- Looking at/Googling for "generic", API-level code is fine, and even encouraged.
- Looking at (or copying) code that has anything at all to do with the assignment you are working on is not OK, even if it's easy to find, unless it was explicitly provided to you in class or in the textbook. If you are using such code, you must cite it properly.
- Discussing the assignments at a high level is OK. Writing any kind of code or pseudocode together is NOT OK.

Diversity and Inclusion

Students with Disabilities

Basic needs and food insecurity

Java Generics

The purpose of generics in Java is to separate data types from algorithms that apply on data types. Generics is in general associated with data structures (CSE 12) and algorithms.

Generic Class

Create a class which has multiple generic methods (like our ArrayList, Set, HashMap classes)

```
public class MyGList<T> {
  private T[] nums;
  public MyGList() {
    nums=null;
  }
  public MyGList(int size) {
    this.nums = (T[])new Object[size];
  }
  public int size() {
    return this.nums.length;
  }
  public void update(int ind, T data) {
    nums[ind] = data;
  }
```

```
  @Override
  public String toString() {
    String str = "";
    for (T n : nums) {
      if (n instanceof Object){
        str += (n.toString() + ", ");
      }
    }
    return str;
  }
public static void main(String[] args){
   MyGList<Integer> vals = new
                  MyGList<Integer>(3);
   System.out.println(vals);
   vals.update(2, 7);
   vals.update(0, 4);
   vals.update(1, 3);
   System.out.println(vals);
  }
}
```

`public interface Collection<E> extends Iterable<`**`E`**`>`

What does the `<E>` mean in the above code?

A. That this collection can only be used with objects of a built-in Java type called `E`
B. That an object reference that implements Collection can be instantiated to work with any object type
C. That a single collection can hold objects of different types

Will the main method compile?

```
public class Generics<T> {
  private ArrayList<T> elements;
  public Generics() {…}
  public T add(T element) {…}
  public int getNumElements() {…}
  public T getLastElement() {…}
  public static void main(String[] args) {
    Generics<T> rr = new Generics<T>();
    rr.add(1);
   System.out.println("Last elem was " + rr.getLastElement());
  }
}
```

A. Yes   B. No

A few notes

You are not allowed to use Generics as follows

- In creating an object of that type:
  ```
  new T() // error
  ```
- In creating an array with elements of that type:
  ```
  new T[100] // error
  ```
- As an argument to instanceof:
  ```
  someref instanceof T // error
  ```
- To ensure that certain methods can be called, we can constrain the generic type to be subclass of an interface or class
  ```
  public class MyGenerics <E extends Comparable>{ ………}
  ```

- Important for data structures in general

```
public class MyList<E>{
     //codes that use E
 }
```

- Type erasure during compile time
  - Compiler checks if generic type is used properly. Then replace them with Object
  - Runtime doesn't have different generic types
  ```
  MyList<String>  ref1 = new MyList<String>();
  MyList<Integer> ref2 = new MyList<Integer>();
  ```
    Compile time

    Runtime

Generics Worksheet

```
import java.io.*;
import java.util.*;
public class GenericWorksheet<E>{
    public E[] data;
    public GenericWorksheet(){

        ________________________________//Initialize data to null
    }
    public Wk1TWorksheet(E[] data){
        //complete a shallow copy
        ________________________________________________

        //what if we want to have a deep copy?
    }
    public E findMedian(){









    }
```

```
    public E findLast(){








    }
    public E removeFirst(){














    }
    public String toString(){










    }
    public static void main(String[] args){
        Integer[] array = new Integer[]{5, 6, 33, 11, 99};

        ___________________________________________________________________
        System.out.println(ref.findMedian());
        System.out.println(ref.findLast());
        System.out.println(ref.removeFirst());
        System.out.println(ref);

       //What if I have another class of my own and want to use it for this
generic class?
    }
}
```