Lab 6: Generics

1.

GenericMaxStack

Time: 30 minutes

Problem Description

You need to create a class named GenericMaxStack that represents a last-in-first-out (LIFO) data structure with the following properties:

1. It has push(int) and pop() operations that work the same way as a normal stack
2. In addition, it has a max() operation that returns the maximum value in the current stack.
3. You have to ensure that your code is working for Integer, Double, and String data types.

Constraints

The max() operation should operate at constant complexity, O(1). This means
you cannot use a loop or recursion to find the minimum value.

Test cases

1. Push 3, 5, 2. Assert max = 5.
2. Push 2, 1, 2, 5. Pop the last element. Assert max = 2. Pop again. Assert max = 2.
3. Push 49.75, 23.54, 100.0. Assert max 100. Pop the last element. Assert max 49.75.
4. Push “OOC is bad”, “Nothing to understand”, and “Try hard”. Assert max “Try hard”. Pop the last element. Assert max “OOC is bad”.

Hint

1. You can use the built-in Stack class if necessary.
2. You can keep up to the max in the stack each time you insert an element in the stack.

5

2.

GenericCount

Time: 40 minutes

Problem Description

Write a generic method to count the number of elements in a list that have a specific property like odd numbers. Keep in mind that this property could be changed into even numbers. So, you should write your code in a way that is open to future changes.

1. Write a class Algorithm which has a generic method **countIf**. This method receives a list of integer and another parameter to know whether you count even or odd.

Test cases

1. Call the count method with a list of numbers 2, 3, 5, 6. Assert 2 odd numbers.
2. Call the countIf method with a list of numbers 2, 3, 16, 6, and even object. Assert 3 even numbers.

5

Hint

1. You have to think about the interface to solve.

3

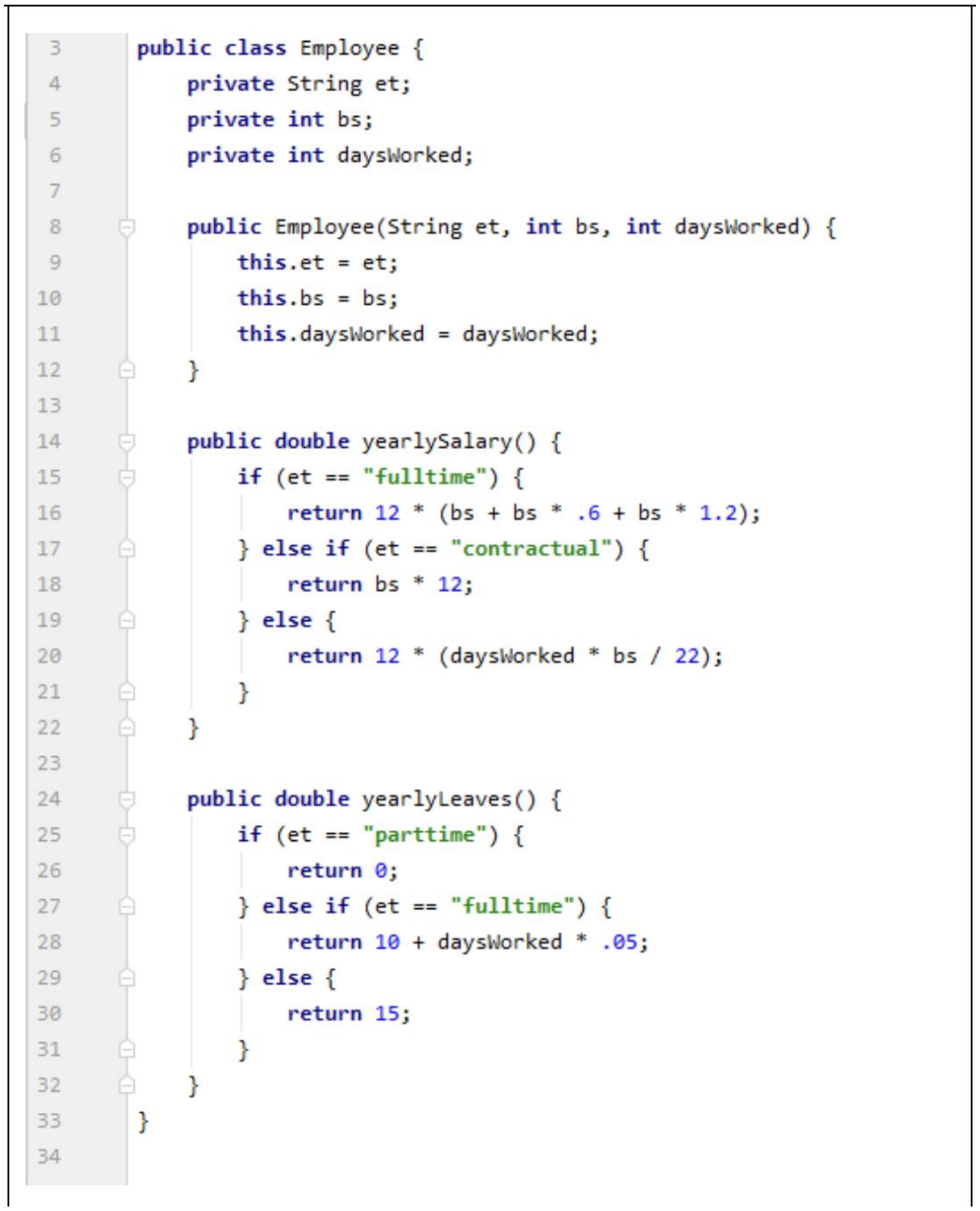
Refactoring

Time: 40 minutes

Refactor the following code:

Test Cases

1. Write 3 test cases for each of the employee types to check their yearly salary and yearly leaves.
2. Write 1 test case to check the type of an object using assertTrue and assertFalse method.



```java
public class Employee {
    private String et;
    private int bs;
    private int daysWorked;

    public Employee(String et, int bs, int daysWorked) {
        this.et = et;
        this.bs = bs;
        this.daysWorked = daysWorked;
    }

    public double yearlySalary() {
        if (et == "fulltime") {
            return 12 * (bs + bs * .6 + bs * 1.2);
        } else if (et == "contractual") {
            return bs * 12;
        } else {
            return 12 * (daysWorked * bs / 22);
        }
    }

    public double yearlyLeaves() {
        if (et == "parttime") {
            return 0;
        } else if (et == "fulltime") {
            return 10 + daysWorked * .05;
        } else {
            return 15;
        }
    }
}
```

5