Quiz 2

Due No due date **Points** 10 **Questions** 20
Available after Apr 19 at 6:30pm **Time Limit** 90 Minutes

Instructions

Let Students See Their Quiz Responses (Incorrect Questions Will Be Marked in Student Feedback)

Only Once After Each Attempt

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	57 minutes	5.5 out of 10 *

* Some questions not yet graded

ⓘ Correct answers are hidden.

Score for this quiz: **5.5** out of 10 *
Submitted Apr 20 at 8:05pm
This attempt took 57 minutes.

Question 1

Not yet graded / 0.5 pts

Explain that Java provides three classes that are helpful when working with strings: **Character**, **String**, and **StringBuilder** and **StringBuffer.**

(minimum of 30 words)

Your Answer:

Java provides three classes when working with
strings: Character, String, and StringBuilder/StringBuffer.

1. Character: It provides various methods to work with individual characters. It used methods include isLetter, isDigit, and toLowerCase . It also used to check whether a character is a letter or a digit or to convert a character to lowercase.

2. String: It represents a sequence of characters. It provides different methods for manipulating strings, including methods for searching, concatenation, comparison, and replacing. In java, Strings are immutable that means that once a string object is created, its value cannot be changed.

3. StringBuilder/StringBuffer: It provides the way to modify strings in a mutable way. Both classes provide methods for appending, inserting, deleting, and replacing characters in a string. The difference between the two classes is that StringBuilder is not thread-safe, while StringBuffer is thread-safe. This means that if multiple threads are modifying the same StringBuffer object, the modifications will be synchronized and will not interfere with each other.

In summary, Character is useful when working with individual characters, String is useful for working with sequences of characters that do not need to be modified, and StringBuilder and StringBuffer are useful when working with mutable strings that need to be modified.

Question 2

0.5 / 0.5 pts

As an object, a String variable name is not a simple data type. It is a(n) ____; that is, a variable that holds a memory address. (Ch 7)

- (●) reference
- ( ) isDigit()
- ( ) StringBuffer
- ( ) StringBuilder

Question 3

Not yet graded / 0.5 pts

Define **concatenation, why we are using? (Ch 7)**

Your Answer:

Concatenation is the process of combining two or more strings and it creates a single string. It involves taking one string and joining another string into it, creating a new longer string that contains the characters of both input strings in sequence.

Question 4

Not yet graded / 0.5 pts

List and describe two mutable string objects in Java.(Ch 7)

Your Answer:

1. StringBuffer:

It is a mutable sequence of characters that can be modified over the time. Because StringBuffer is mutable, it is useful when we need to modify the contents of a string frequently, such as when constructing a long string from smaller components The StringBuffer class provides methods for inserting, appending, deleting, and replacing characters in the string. It is thread-safe, which means that can be safely accessed by multiple threads simultaneously without the risk of data corruption.

2. StringBuilder:

This is similar to StringBuffer, but it is not threadsafe. It gives similar methods for modifying the contents of a string, such as append() , insert() , delete() , and replace(). It is faster than StringBuffer because it does not provide thread-safety guarantees.

The StringBuffer and StringBuilder gives methods to convert the mutable string to an immutable String object, such as toString(). This allows us to use the mutable string object for efficient string manipulation and then convert it back to an immutable String object when we are finished modifying it.

Question 5

Not yet graded / 0.5 pts

Describe of using an **anonymous object (ch 7) (minimum of 30 words)**

Your Answer:

In Java, an anonymous object is an object that is created without assigning it to a variable or giving it a name. Instead, it is created and used in a single statement or expression. Anonymous objects are commonly used when we need to create a temporary object for a specific purpose and do not need to reuse it elsewhere in the program. One common use case for anonymous objects is when invoking a method that expects an object as an argument.

For example,
suppose we have a method doSmtg that takes a Person object as an argument:

```
public void doSmtg(Person a) {
 // do some with the Person object
}
```

If we only need to pass a Person object to this method once and do not need to reuse it elsewhere, we can create an anonymous object and pass it directly to the method:

```
doSmtg(new Person("Lusi", "Mala"));
```

In this example, we create an anonymous Person object with the name "Lusi" and the last name "Mala" and pass it directly to the doSmtg method. The anonymous object is not assigned to a variable or given a name, but it is still a fully functional object that can be used by the method.

We can also do calculations with anonymous method.

Question 6

Not yet graded / 0.5 pts

- Declare and use arrays of objects (Ch 8) (minimum of 30 words)

Your Answer:

An array of objects is a data structure that allows storing multiple instances of a class or a struct in a contiguous memory block. To declare an array of objects, we can use the following syntax:

```
class Dharmi {

// class definition
};
```

Dharmi objects[5]; // an array of 5 Dharmi objects
This declares an array of 5 Dharmi objects. We can access individual objects in the array using their index, like objects[0] or objects[4].
To use the array, we can initialize its elements using a loop or by assigning values directly to each element:

```
for (int i = 0; i < 5; i++) {
   objects[i] = Dharmi(); // initialize each object with the default constructor
}

objects[0].method(); // call a method on the first object
```

In this, we initialize each object in the array with the default constructor Dharmi and then call a method on the first object using the array index notation. We can use arrays of objects to store collections of related data, such as a list of students or employees in a company.

Question 7

Not yet graded / 0.5 pts

Explain how to initialize an array by assigning values to individual array elements or to the entire array using curly brackets.(Ch 8) minimum of 30 words

Your Answer:

In Java, one can initialize an array by assigning values to individual array elements using curly brackets. To assign values to individual elements, can use the array index notation:

```
int[] array = new int[2];
array[0] = 5;
array[1] = 10;
```

This creates an array of size 2and assigns the values 5 and 10 to its elements at index 0 and 1 respectively.
We can use the curly bracket notation to initialize the entire array:

```
int[] array= {10, 20};
```

This creates the same array as before.

Question 8

0.5 / 0.5 pts

You declare an array variable in the same way you declare any simple variable, but you insert a pair of curly brackets after the type.(Ch 8)

- ○ True
- ◉ False

Question 9

0.5 / 0.5 pts

The last subscript in an array of size 100 is ____.(Ch 8)

- ○ 102
- ○ 97
- ◉ 99
- ○ 101

Question 10

Not yet graded / 0.5 pts

Define the term **parallel array** (Ch 8)

Your Answer:

A parallel array is a programming concept in which two or more distinct arrays are used to represent connected data in such a way that their corresponding components are related to each other by their indices.

Parallel arrays are useful for storing and processing related data with diverse data types or structures. Because we can access the associated elements of the arrays using their indices, they allow us to conduct operations on the linked data in a simpler and more efficient manner. However, when working with arrays of varying sizes or attempting to preserve the link between the arrays, parallel arrays can inject complexity into the code.

Question 11

0.5 / 0.5 pts

Searching an array for an exact match is not always practical.(Ch 8)

- (●) True
- ( ) False

Question 12

0.5 / 0.5 pts

Arrays, like all nonprimitive objects, are ____ types; this means that the object actually holds a memory address where the values are stored, and the receiving method gets a copy of the array's actual memory address.

- ( ) concatenation
- ( ) range match
- (●) reference

Question 13

0.5 / 0.5 pts

You declare an array variable in the same way you declare any simple variable, but you insert a pair of curly brackets after the type.(Ch 8)

- ( ) True
- (●) False

Question 14

0.5 / 0.5 pts

A(n) ____ is an integer contained within square brackets that indicates one of an array's variables, or elements.(Ch 8)

- (●) subscript
- ( ) an array
- ( ) parallel array

Question 15

0.5 / 0.5 pts

A class diagram consists of a rectangle divided into five sections.(Ch 10)

- ( ) True
- (●) False

Question 16

Not yet graded / 0.5 pts

Define the term **subtype polymorphism**.(Ch 10)

Your Answer:

Subtype polymorphism is a key concept in object-oriented programming because it allows objects of distinct classes to be considered as instances of a single superclass or interface. Subtype polymorphism is a strong tool for building reusable and extendable code, and it is one of the primary advantages of utilizing an object-oriented programming paradigm.

A subclass can be replaced for its superclass or interface type in subtype polymorphism, and the program's behavior remains unaltered. This allows us to develop more general code that can operate with

objects from numerous related classes without having to know the exact implementation details of each class.

Question 17

0.5 / 0.5 pts

Using the same method name to indicate different implementations is called polymorphism.(ch 10)

- (●) True
- ( ) False

Question 18

0.5 / 0.5 pts

When you instantiate an object that is a member of a subclass, you are actually calling at least two constructors.(Ch 10)

- (●) True
- ( ) False

Question 19

0.5 / 0.5 pts

The keyword super always refers to the superclass of the class in which you use it.(ch 10)

- (●) True
- ( ) False

Question 20

Not yet graded / 0.5 pts

Describe how Java uses **virtual method calls**.(ch 10)

Your Answer:

In Java, all non-static methods are virtual methods. This implies that when a method is called on an object, the runtime system will select which implementation of the method to use based on the actual type of the object at runtime, rather than the stated type of the variable that refers to the object.

Assume we have a superclass music with a method music() that outputs "rock" and a subclass pop that overrides music() to print "pop". If we create a new rock object and use its music() method, the Java runtime system will automatically execute the rock class's music() function, even if the reference variable used to refer to the object is of the music type:
music a = new rock(); a.music(); // prints pop

Virtual method calls are an essential element of object-oriented programming because they allow code to be written in a more flexible and extendable manner. We may design code that interacts with objects of multiple kinds by utilizing virtual method calls, which eliminates the requirement to know the exact type of each object at compile time. This makes it easy to build reusable code that can be extended and updated in the future.

Quiz Score: **5.5** out of 10