

Midterm Test 2

! This is a preview of the published version of the quiz

Started: Dec 14 at 11:19pm

Quiz Instructions

- This is an online test.
 - Exam Time: Dec 8, 12:10 - 14:40 (additional 10 min to set up your IDE + 2.5 Hr Test)
 - Once you finish solving the exam, please make sure to submit your answers through Canvas.
 - The submission page will be closed after the due date & time, so make sure to finish the submission in time.
-
- Please double check that JAVA JDK and your IDE is properly installed on your computer.
 - Once it is confirmed, please start the exam from 12:10 PM.
 - You are allowed to make up to two multiple attempts and the highest score will be kept.



Question 1

2 pts

In Java, which of the following is true regarding inheritance?

- ☐ Java does not support multiple inheritance but allows it through interfaces.
- ☐ Java supports multiple inheritance through classes
- ☐ Java only allows multiple inheritance through abstract classes.
- ☐ Multiple inheritance is not possible in Java, neither through classes nor interfaces.



Question 2

2 pts

Which statement is correct regarding object typecasting in Java?

- ☐ Java allows both upcasting and downcasting of objects, but downcasting is limited.
- ☐ Typecasting of objects is not supported in Java.
- ☐ Upcasting in Java refers to casting a parent class object to a child class object.
- ☐ Objects in Java can be typecast only through downcasting.



Question 3

2 pts

In Java, is it possible to override a private method? Select the correct answer.

- ☐ No, private methods cannot be overridden because they are not accessible in the child class.
- ☐ Yes, private methods can be overridden, but only if they are declared as static.
- ☐ Yes, private methods can be overridden just like public methods.
- ☐ No, private methods can be accessed but not overridden in the child class.



Question 4

2 pts

Which of the following statements is true about constructors in Java?

- ☐ Constructors can be declared as final to prevent inheritance.
- ☐ Constructors implicitly return the current instance of the class and can be overloaded.
- ☐ Constructors explicitly return a value and can have a return type specified.
- ☐ Constructors cannot be overloaded under any circumstances.



Question 5

2 pts

Which statement is correct regarding static members in Java?

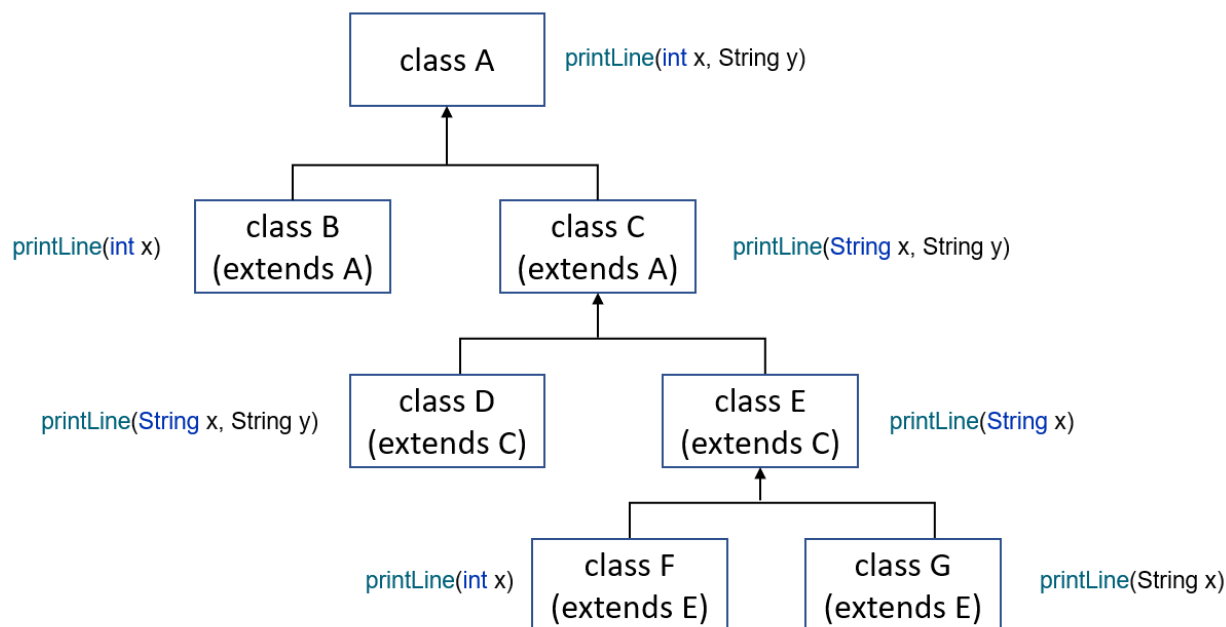
- ☐ Static methods can override non-static methods and use 'this' and 'super' keywords.
- ☐ Static variables are unique to each object and are not memory efficient.
- ☐ Static variables and methods belong to the class, and static methods cannot access non-static members directly.
- ☐ Static methods can be overridden, and static members cannot be declared in an abstract class.



Question 6

5 pts

Given the following class diagram, answer which class's printLine() method will be used for each statement below.



Assume z is an object instantiated from class F as follows; `F z = new F();`

- 1) `z.printLine(1)` will use `printLine()` method from class
- 2) `z.printLine(2, "test")` will use `printLine()` method from class
- 3) `z.printLine("test")` will use `printLine()` method from class
- 4) `z.printLine("mid-term", "test")` will use `printLine()` method from class



Question 7

4 pts

In the following code, Class BBB inherits Class AAA, which includes private field members.

```
class AAA {  
    1 usage  
    private int a;  
}  
  
2 usages  
class BBB extends AAA {  
}  
  
public class exCase01 {  
  
    public static void main(String[] args) {  
  
        BBB objB = new BBB();  
  
        AAA objA = new AAA();  
  
        System.out.println(objB.a);    }  
}
```

However, the given code causes an error as follows

java: a has private access in AAA

Explain why this error occurs and state how we can fix the code to print the value of the private variable **a**.

Edit View Insert Format Tools Table

12pt Paragraph **B** *I* U A T^2

p

0 words



Question 8

2 pts

What is the correct representation of 'this' keyword functionality?

- ☐ 'this' can be used to invoke methods from a superclass.
- ☐ 'this' can only be used to refer to static variables within a class.
- ☐ 'this' is used exclusively for invoking static methods of the current class.
- ☐ 'this' is used to refer to the current class instance variable, invoke current class methods, constructors, and can be passed as an argument or returned from methods.

**Question 9****2 pts**

Which of the following statements is true regarding method overriding in Java?

- ☐ Static methods can be overridden because they are associated with objects.
- ☐ Overloaded methods cannot be overridden in any scenario.
- ☐ Static methods cannot be overridden, but overloaded and non-private methods can be.
- ☐ Private methods can be overridden as long as they are accessible within the same class.

**Question 10****4 pts**

Consider the following code.

```
class Parent1 {  
    2 usages  
    int myInt;  
  
    1 usage  
    Parent1() {        myInt = 25;    }  
}
```

2 usages

```
class Child1 extends Parent1 {  
    1 usage  
    int myInt;  
    1 usage  
    Child1() {  
        // Code block <- Change here  
    }  
}
```

```
1 usage  
void display() {  
    System.out.println("Value of myInt in the super class is: " + super.myInt);  
    System.out.println("Value of myInt in the child class is:" + myInt);    }  
}
```

```
class Demo1 {  
    public static void main(String[] args) {  
        Child1 childObject = new Child1();  
        childObject.display();    } }  
}
```

We want the code output to be as follows (You may ignore the bullet symbol)

- Value of myInt in the super class is: 25
- Value of myInt in the child class is:0

Modify the Child1 constructor appropriately to achieve the given output.

Edit View Insert Format Tools Table

12pt Paragraph **B** *I* U A    τ^2 

p

  | 0 words |   



Question 11

4 pts

Consider the following code. We expect 6 to be printed by creating an object with Demo2B(1, 5).

However, the given code includes both correct and incorrect statement between line 8-11, which causes an error.

```
class Demo2A {
    public Demo2A(int x) {
        System.out.print(x);
    }
}

class Demo2B extends Demo2A {
    public Demo2B(int a, int b) {
        int c = a + b;           // line 8
        super(c);                // line 9

        super(a + b);           // line 11
    }
}

public class Demo2 {
    public static void main(String[] args) {
        Demo2B temp1 = new Demo2B(1, 5);
    }
}
```

Explain which code statement should be deleted and which statement should be kept.

Edit View Insert Format Tools Table

12pt Paragraph **B** *I* U **A**

p

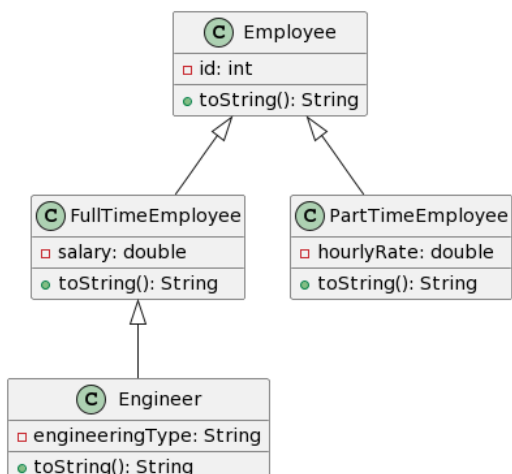
| 0 words |



Question 12

8 pts

Implement the following diagram with four classes into a Java source code.



- Employee has an integer type data field 'id' and two methods, one constructor and a method toString that returns a String "An employee with ID " + id.
- Employee class is the superclass, which has two subclasses, a FullTime class and a PartTime class. Engineer class is inherited by the FullTime class.
- FullTime class inherits the id field from the parent class Employee and has one additional data member of its own, salary of type double. The id is also inherited by the PartTime class. Two more data members, hourlyRate and hoursWorked of type double, are declared in the PartTime class to determine the compensation. Both subclasses have a method toString which is an overriding method of the one in the Employee class.
- Engineer class inherits the FullTime class and Includes a data member which describes the type of engineering and a method toString.

Once the JAVA class is finished, you may use the following main() method to test the program.

```
public class Main {  
    public static void main(String[] args) {  
        Employee employee1 = new Employee(1);  
        FullTimeEmployee fullTimeEmployee1 = new FullTimeEmployee(2, 50000.0);  
        PartTimeEmployee partTimeEmployee1 = new PartTimeEmployee(3, 20.0, 40.0);  
        Engineer engineer1 = new Engineer(4, 75000.0, "Software Engineering");  
  
        System.out.println(employee1);  
        System.out.println(fullTimeEmployee1);  
        System.out.println(partTimeEmployee1);  
        System.out.println(engineer1);  
    }  
}
```

Edit View Insert Format Tools Table

12pt Paragraph B I U A v P v T² v | :

p

  | 0 words |   



Question 13

7 pts

Consider the following code. To use it, please import the following packages.

```
import java.util.Arrays;  
import java.util.Collections;
```

```
public class Demo3 {  
  
    public static void main(String args[]) {  
  
        Integer[] arr = {5, -2, 23, 7, 87, -42, 509};  
        Integer[] arr_downshift = new Integer[arr.length];  
        Integer[] arr_ascending = new Integer[arr.length];  
        Integer[] arr_descending = new Integer[arr.length];  
  
        System.out.println("The original array is: ");  
        for (int num : arr) {  
            System.out.print(num + " ");  
        }  
        System.out.println(" ");  
  
        // Sort array downshifting by 1, then  
        // Sort the elements of an array in ascending order, then  
        // sort the elements of an array in descending order  
    }  
}
```

Modify the code by downshifting the array by 1 , then sort ascending, sort descending.

You may copy the original ArrayList into 3 different arrays, then sort the copied array.

Once you perform any kind of sorting, please make sure to print it on the screen.

Edit View Insert Format Tools Table

12pt Paragraph B I U A T²

p

0 words </> ↗ ⋮



Question 14

2 pts

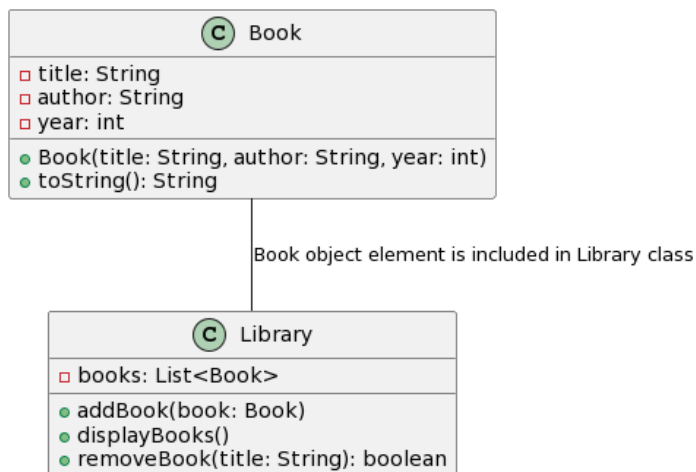
Explain the difference between checked and unchecked exceptions in Java. Provide any example of each exceptions and discuss when each exceptions occur.

**Question 15****2 pts**

Describe the purpose of the "finally" block in Java's exception handling. Provide a specific scenario where the "finally" block is required.

**Question 16****15 pts**

You are supposed to implement a Book Management System in Java based on the following UML diagram.



The system is composed of two classes: **Book** and **Library**.

1. Class: Book

- Attributes:
 - **title**: String (private)
 - **author**: String (private)
 - **year**: int (private)
- Methods:
 - **Book(title: String, author: String, year: int)**: Constructor to initialize the book with the provided title, author, and publication year.
 - **toString(): String**: Method to return a formatted string representation of the book.

2. Class: Library

- Attributes:

- `books`: `ArrayList<Book>` (private)
- Methods:
 - `addBook(book: Book)`: Method to add a book to the library's collection.
 - `displayBooks()`: Method to display all books in the library.
 - `removeBook(title: String): boolean`: Method to remove a book from the library based on its title. Returns true if the book is successfully removed, false otherwise.

Furthermore, you may use the following `main()` method to test the code that you wrote.

```
public class BookManagementSystem {
    public static void main(String[] args) {
        Library library = new Library();

        // Adding sample books
        library.addBook(new Book("Java Programming", "John Doe", 2020));
        library.addBook(new Book("Data Structures", "Jane Smith", 2018));
        library.addBook(new Book("Electrical Engineering Basics", "Alice Johnson", 2022));

        // Displaying books
        library.displayBooks();

        // Removing a book
        library.removeBook("Java Programming");

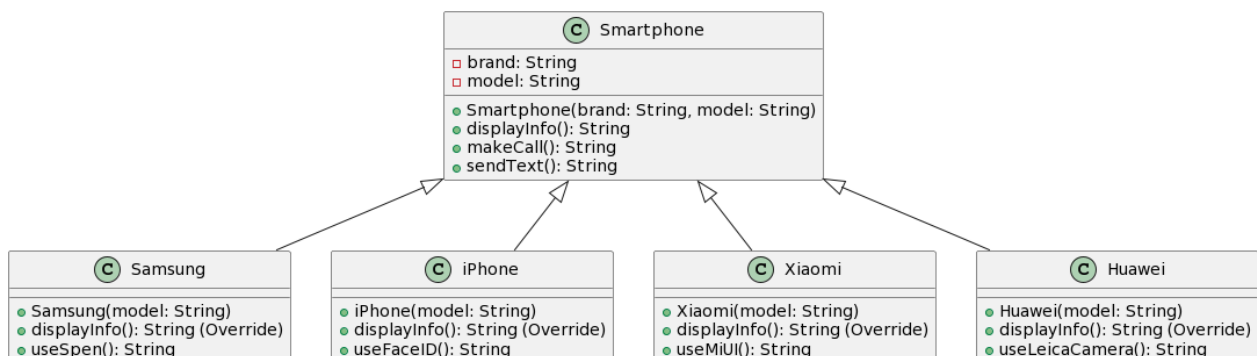
        // Displaying books after removal
        library.displayBooks();
    }
}
```



Question 17

15 pts

You are tasked to implement a hierarchy of smartphone classes in Java, that include `Smartphone`, `Samsung`, `iPhone`, `Xiaomi`, and `Huawei`. Each class has specific attributes and methods, showcasing the concept of specialization and method overriding.



(If the image is not clear, right-click it and open the image in a new tab)

1. Implement the `Smartphone` class with private attributes `brand` and `model`. Include a constructor and methods: `displayInfo()`, `makeCall()`, and `sendText()`.
2. Implement the child classes `Samsung`, `iPhone`, `Xiaomi`, and `Huawei`, each extending the `Smartphone` class. The constructor of each child class should initialize the `model` attribute, and the `displayInfo()` method should override the one in the parent class to provide specific brand and model information.
3. Introduce specific methods for each brand showcasing their specialties:
 1. In `Samsung`, implement `useSpen()` that prints "Using the S Pen feature."
 2. In `iPhone`, implement `useFaceID()` that prints "Using Face ID for authentication."
 3. In `Xiaomi`, implement `useMiUI()` that prints "Navigating the MIUI interface."
 4. In `Huawei`, implement `useLeicaCamera()` that prints "Capturing photos with Leica Camera technology."

You may write your own `main()` method, or use the following code.

```
public class SmartPhoneDemo {
    public static void main(String[] args) {
        Samsung samsungPhone = new Samsung("Galaxy S21");
        iPhone iPhoneX = new iPhone("X");
        Xiaomi xiaomiPhone = new Xiaomi("Redmi Note 10");
        Huawei huaweiPhone = new Huawei("P40 Pro");

        // Display information and demonstrate specific functionalities
        displayAndDemonstrate(samsungPhone);
        displayAndDemonstrate(iPhoneX);
        displayAndDemonstrate(xiaomiPhone);
        displayAndDemonstrate(huaweiPhone);
    }

    private static void displayAndDemonstrate(Smartphone smartphone) {
        System.out.println("Smartphone Info: " + smartphone.displayInfo());
        System.out.println("Make a Call: " + smartphone.makeCall());
        System.out.println("Send a Text: " + smartphone.sendText());

        if (smartphone instanceof Samsung) {
            System.out.println(((Samsung) smartphone).useSpen());
        } else if (smartphone instanceof iPhone) {
            System.out.println(((iPhone) smartphone).useFaceID());
        } else if (smartphone instanceof Xiaomi) {
            System.out.println(((Xiaomi) smartphone).useMiUI());
        } else if (smartphone instanceof Huawei) {
            System.out.println(((Huawei) smartphone).useLeicaCamera());
        }

        System.out.println(); // Add a newline for better readability
    }
}
```




Question 18

10 pts

In the following, you are given the abstract `Book` class and a `Main` class. `MyBook` Class extends the abstract `Book` class and currently, it is blank. Your task is to fill out the `MyBook` class.

In the `Main` class, we created an object of the `MyBook` Class.

```
abstract class Book{
    2 usages
    String title;
    1 usage 1 implementation
    abstract void setTitle(String s);
    1 usage
    String getTitle(){
        return title;
    }
}

2 usages
class MyBook extends Book {
    1 usage
    
}

public class MyBookExample {
    public static void main(String[] args) {
        MyBook myBook = new MyBook();
        myBook.setTitle("A tale of two cities");

        System.out.println("The title is: " + myBook.getTitle());
    }
}
```

For a sample input "A tale of two cities", you are supposed to get "The title is: A tale of two cities" as the output.

**Question 19****10 pts**

Did you finished the TLQ?

If not, please fill it out from the following link.

<https://onlinesurvey.cityu.edu.hk/tlq> (<https://onlinesurvey.cityu.edu.hk/tlq>)

☐ True

☐ False

