ISE-2 JAVA

Registration.no: (2022BIT031)
Name:Utkarsh Vinayak Kshirsagar
Batch: [D]

Q.a) Imagine you are tasked with designing a system for Library Management System managing books and their borrowing records. You have a Book class to represent each book and a Library class to oversee book management.

Determine appropriate interfaces if any along with member methods and variables, specifying their data types and modifiers. Explain the purpose and rationale behind your choices of data types and modifiers for each member in these classes.

Ans:

Book Class:

```
   // Book class to represent each book
public class Book {
   private String title;
   private String author;
   private int yearPublished;
   private boolean isAvailable;

   // Constructor to initialize a book with title, author, and yearPublished
   public Book(String title, String author, int yearPublished) {
      this.title = title;
      this.author = author;
      this.yearPublished = yearPublished;
      this.isAvailable = true; // Initially, book is available
   }

   // Getter and setter methods for title, author, yearPublished, and isAvailable
   // Getter methods provide read access to private member variables
   // Setter methods provide write access to private member variables

   // Getters
   public String getTitle() {
```

```
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public int getYearPublished() {
        return yearPublished;
    }

    public void setYearPublished(int yearPublished) {
        this.yearPublished = yearPublished;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    // Method to set the availability of the book
    public void setAvailable(boolean available) {
        isAvailable = available;
    }
}
```

Library Class:

```
import java.util.ArrayList;

// Library class to oversee book management
public class Library {
    private ArrayList<Book> books;
```

```
// Constructor to initialize an empty list of books
public Library() {
    this.books = new ArrayList<>();
}

// Method to add a book to the library
public void addBook(Book book) {
    books.add(book);
}

// Method to remove a book from the library
public void removeBook(Book book) {
    books.remove(book);
}

// Method to display all available books in the library
public void displayAvailableBooks() {
    for (Book book : books) {
        if (book.isAvailable()) {
            System.out.println(book.getTitle() + " by " + book.getAuthor());
        }
    }
}

// Method to borrow a book from the library
public void borrowBook(Book book) {
    if (book.isAvailable()) {
        book.setAvailable(false);
        System.out.println("You have borrowed " + book.getTitle());
    } else {
        System.out.println("Sorry, the book is not available for borrowing.");
    }
}

// Method to return a borrowed book to the library
public void returnBook(Book book) {
    book.setAvailable(true);
    System.out.println("You have returned " + book.getTitle());
}
```

}

Q.b) Explain the significance of the error message encountered during the compilation of the

ISEII.java code: "ISEII.java:20: error: unreported exception InterruptedException; must be caught or declared to be thrown Thread.sleep(1213);"

Also describe all potential solutions to address this error.

Ans:

The error message we encountered during the compilation of the `ISEII.java` code is indicating that there's an unchecked exception, specifically `InterruptedException`, which is being thrown within the code block where `Thread.sleep(1213)` is called. The message is saying that this exception either needs to be caught using a try-catch block or declared to be thrown using the `throws` clause in the method signature.

Here's a breakdown of the error message:

- ISEII.java:20: This specifies the file (`ISEII.java`) and line number (line 20) where the error occurred.
- error: unreported exception InterruptedException: This part of the error message is stating that an `InterruptedException` is being thrown.
- must be caught or declared to be thrown: This indicates that you need to handle this exception somehow, either by catching it with a try-catch block or by declaring it in the method signature.

Now, let's discuss potential solutions to address this error:

1. Use a try-catch block: You can surround the `Thread.sleep(1213)` call with a try-catch block to catch the `InterruptedException` and handle it appropriately. For example:

```
java
try {
    Thread.sleep(1213);
} catch (InterruptedException e) {
    // Handle the interruption
    e.printStackTrace();
```

```
}
```

2. Throw the exception: If the method where this code resides is not suitable for handling the `InterruptedException`, you can declare that the method throws this exception. This means that any method calling this method would need to handle or declare this exception as well. For example:

```
java
public void myMethod() throws InterruptedException {
    Thread.sleep(1213);
}
```

3. Handle the interruption appropriately: Depending on the context of your code, you might want to handle the interruption in a specific way. This could involve logging the interruption, retrying the operation, or stopping the thread gracefully.

Remember to choose the solution that best fits the requirements and design of your program.