1. Here is part of the UML diagram for the attached code. Complete the diagram.

**Note: For this quiz only**, you can use bold text instead of italic text if you have difficulty writing text that is clearly italic.

```
<< Abstract >>

Book
- title: String
- itemGroup: String

+ Book()
Book(title: String)
+ getTitle(): String
+ getCreators(): String
# getItemGroup(): String

<< Interface >>

Shelf

+ addLeft(shelfItem: T): void
+ addRight(shelfItem: T): void
+ getLeft(): T
+ getRight(): T

<< Abstract >>

Book
- title: String
- itemGroup: String

+ Book()
Book(title: String)
+ getTitle(): String
+ getCreators(): String
# getItemGroup(): String

(insert dotted upwards arrow here)

BookShelf
- contents: LinkedList<T>

+ BookShelf()
+ BookShelf(item: T)
+ addLeft(item: T): void
+ addRight(item: T): void
+ getLeft(): T
+ getRight(): T

<< Abstract >>

TextBook
- authors: ArrayList<String>
# edition: int

+ TextBook()
+ TextBook(title: String, authors: ArrayList<String>)
+ getCreators(): String

ComicBook
- author: String
- artist: String

+ ComicBook()
+ ComicBook(title: String, author: String)
+ ComicBook(title: String, author: String, artist: String)
+ getCreators(): String

(insert two solid upwards arrows here)
2.(a) Write a main method that creates a BookShelf of Strings, where each String is the title of one of the following books: “It”, “Cujo”, and “The Shining”. Your code should add each String to the BookShelf in the order they appear on the list. However, the BookShelf’s contents should store them in alphabetical order.

Also create a BookShelf that stores objects of type Book. It contains a TextBook called “Introduction to Algorithms” by “Cormen” and “Leiserson”.

One possible solution:

```java
public static void main(String[] args) {
    BookShelf<String> demoStringBookShelf = new BookShelf<String>("It");
    demoStringShelf.addLeft("Cujo");
    demoStringShelf.addRight("The Shining");

    ArrayList<String> algAuthors = new ArrayList<String>();
    algAuthors.add("Cormen");
    algAuthors.add("Leiserson");
    TextBook algBook = new Textbook("Introduction to Algorithms", algAuthors);
    BookShelf<Book> demoBookShelf = new BookShelf<Book>(algBook);
}
```

2.(b) Consider the following code contained in a main method that is not in any of the given classes:

1. ComicBook cm1 = new ComicBook("SuperHero", "Comic Author");
2. Shelf<Object> shelf0 = new BookShelf<Object>(cm1);
3. System.out.println((shelf0.getLeft()).getCreators());

This code will prevent the program from running. Explain the problem and suggest two different ways to fix it. Refer only to these three lines of code and the supplementary code at the end of this booklet.

Solution:

On line #3, the method getLeft() will return an value of type Object which does not have a getCreators() method. This can be fixed by either:

(i) changing line #2 to say: Shelf<ComicBook> shelf0 = new BookShelf<ComicBook>(cm1); or
(ii) casting on line #3: System.out.println(((ComicBook) shelf0.getLeft()).getCreators());
3.(a) Write the code for a class called **DishRack** that implements interface **Shelf** and a class called **Dish** that can be stored in a **DishRack**. A **DishRack** can store a maximum of 20 instances of **Dish**. It should not be a generic class; instead, it should always only contain **Dish** objects. Class **Dish** contains a **String** called **colour** and a **String** called **size**, a constructor, methods for getting and setting **colour**, getting (but not setting) **size**, and a **toString()** method that returns the colour and size together in a **String** such as "medium green". You do **not** need to write a **main** method for either class.

```java
public class Dish {
    private String colour;
    private String size;

    public Dish(String size) {
        this.size = size;
    }

    public String getColour() {
        return colour;
    }

    public String getSize() {
        return size;
    }

    public void setColour(String colour) {
        this.colour = colour;
    }

    public String toString() {
        return size + " " + colour;
    }
}
```

```java
public class DishRack implements Shelf<Dish> {
    private LinkedList<Dish> contents = new LinkedList<Dish>();

    public DishRack() {
    }

    public DishRack(Dish dish) {
        contents.add(0, dish);
    }

    public void addLeft(Dish dish) {
        if (contents.size < 20) {
            contents.add(0, dish);
        } else {
            System.out.println("Dish Rack is full.");
        }
    }
}
```
public void addRight(Dish dish) {
    if (contents.size < 20) {
        contents.add(dish);
    } else {
        System.out.println("Dish Rack is full.");
    }
}

public Dish getLeft() {
    return contents.get(0);
}

public Dish getRight() {
    return contents.get(contents.size() - 1);
}

3.(b) What are two big differences between class DishRack and BookShelf?

DishRack has a fixed capacity while Bookshelf does not.

DishRack only stores variables of type Dish, while BookShelf is generic and can store any type.

3.(c) Why is Book an abstract class and not an interface?

Class Book contains methods that are not static, but have been implemented. This is allowed for an abstract class, but not an interface.
public interface Shelf<T> {

    /** Adds shelfItem to the beginning of the shelf. */
    public void addLeft(T shelfItem);

    /** Adds shelfItem to the end of the shelf. */
    public void addRight(T shelfItem);

    /** Returns the left-most item on the shelf. */
    public T getLeft();

    /** Returns the right-most item on the shelf. */
    public T getRight();
}

import java.util.LinkedList; // LinkedList replaced ArrayList

public class BookShelf<T> implements Shelf<T> {

    private LinkedList<T> contents = new LinkedList<>();

    public BookShelf() {
    }

    public BookShelf(T item) {
        contents.add(0, item);
    }

    public void addLeft(T item) {
        contents.add(0, item);
    }

    public void addRight(T item) {
        contents.add(item);
    }

    public T getLeft() {
        return contents.get(0);
    }

    public T getRight() {
        int numBooks = contents.size();
        return contents.get(numBooks - 1);
    }
}

The questions in the comments of the following code are not for marks.
public abstract class Book {

    private String title;
    protected static String itemGroup = "Book"; // itemGroup used to be public.

    public Book() {
    }

    public Book(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }

    public abstract String getCreators();

    protected static String getItemGroup() {
        return itemGroup;
    }
}

import java.util.ArrayList;

public class TextBook extends Book {

    private ArrayList<String> authors = new ArrayList<>();
    protected int edition = 1;

    public TextBook() {
    }

    public TextBook(String title, ArrayList<String> authors) {
        super(title);
        this.authors = authors; // This points to the parameter ArrayList instead of copying it.
        // What happens if the parameter list is changed elsewhere?
    }

    public void setEdition(int ed) {
        if(ed >0) {
            edition = ed;
        }
    }

    public String getCreators() {
        return authors.toString();
    }
}
public class ComicBook extends Book {

    private String author;
    private String artist;

    public ComicBook() {
    }

    public ComicBook(String title, String author) {
        this(title, author, null);
    }

    public ComicBook(String title, String author, String artist) {
        super(title);
        this.author = author;
        this.artist = artist;
    }

    public String getCreators() {
        return "Author: " + author + ", Artist: " + artist;
    }
}

If you remove the code from the booklet, be sure to **write your name** on the back of the remaining pages.