6. Menüs

Aufgaben:

Implementieren Sie den Menüpart eines Programms. Dabei soll es ein Menü namens „Player“ geben. Es sollen Spieler über das Item „add Player“ hinzugefügt und über „remove Player“entfernt werden. Mit *JoptionPane.showInputDialog(Component parent, String message)* können sie einen Dialog öffnen, damit der User jeweils den Namen des Spielers angibt. Die Methode *showInputDialog* gibt einen String zurückgibt.

```
public MyPlayerMenu() {

	ArrayList<Player> player = new ArrayList<>();









	setSize(300, 300);
	setDefaultCloseOperation(EXIT_ON_CLOSE);
	setVisible(true);
}

public class Player{

	private String name;

	public Player (String name) {
		this.name = name;
	}

	public String getName() {
		return name;
	}
}
```

7. Graphics und Timer

Aufgabe:

Ein Rechteck soll sich nach links bewegen.
Erzeuge einen Timer und änder entsprechend die x-Koordinate des Rechtecks.
Was darfst du auf keinen Fall vergessen, damit man die Änderung auch sehen kann?

```
private Rectangle rect;

public MovingRect() {









        setSize(300, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
}

private class MyPanel extends JPanel{

        @Override
        protected void paintComponent(Graphics g) {
                super.paintComponent(g);







        }
}
```

8. Threads

Aufgabe:

Ihr seid als Backendprogrammierer verantwortlich, dass das Buchungssytem eines Fluganbieters einwandfrei funktioniert. Dabei gibt ein eine Klasse Flugbuchung, die eine ArrayListe von Flügen hat.
Ein Flug hat u.a. ein Datum, eine Startort, einen Ankunftsort. Schreiben Sie in der Klasse Flug eine Methode equals, die diese Attribute überprüft.
In der Klasse Flugbuchung soll eine Methode buchen sein, die bei dem Flug Objekt die Methode reservieren aufruft und so einen Platz reserviert. Sorgen Sie dafür, dass nur ein Benutzer zur Zeit auf die buchen()-Methode zugreifen kann.

```
public class Flug {

	private Date zeit;
	private String startFH;
	private String zielFH;
	private int freieSitze;

	//hier gibt es noch weitere Methoden wie Getter und  einen Konstruktor

	public boolean reservieren (int sitze) {
		if (freieSitze >= sitze) {
			freieSitze -= sitze;
			return true;
		}
		return false;
	}

	@Override
	public boolean equals(Flug other) {









	}
}


public class Flugbuchung {

	private ArrayList<Flug> fluege;

	public Flugbuchung() {
		fluege = new ArrayList<>();

		//hier wird noch die ArrayListe befüllt
	}
```

```
        //TODO: Schreiben Sie hier eine Methode buchen (Flug flug, int anzahlSitze), die
        true zurückgibt, wenn die Buchung erfolgreich war und sonst false. Verwenden Sie
        dafür die Methode reservieren aus der Klasse Flug.
```

```
}
```

9. Sockets

Aufgabe:
Ein User soll wie bei Whatsapp eine Nachricht an einen anderen User schicken können. Dabei wird als erstes die ID (String) des Absenders, dann die ID des Empfängers und dann die Nachricht versendet. Der Server hat eine Hashmap <String, Socket> users, in denen jeder ID ein Socket zugeordnet sein kann.
Schreibe die Methode receive, die eine Nachricht empfängt und sie an den entsprechenden User weiterleitet.

```java
public class Server {

	private HashMap<String, Socket> user = new HashMap<>();

	public Server() {
		//hier wird die Verbindung zu dem User aufgebaut etc.
	}

	public void empfangen(Socket socket) {
		String from = "";
		String to = "";
		String message = "";
		PrintWriter writeToOther;
		Scanner reader;

		while (isRunning) {
			try {









			} catch (Exception e) {

			}
		}
	}
}
```

Fragen:

Was ist ein Serversocket? Was kann der Server damit machen? Ein ServerSocket hat zwei Parameter, wofür stehen diese?

Über welche Schnittstelle kommunizieren Client und Server mit einander?