# Lecture Notes

THURSDAY, FEBRUARY 13, 2025

# Lecture Notes

## Introduction to Tuples Conference

- Tuples is a student conference organized by Type Sig, a student society on programming languages and theoretical computer science.
- The conference features 12 speakers from across the UK, covering topics in theoretical computer science and programming languages.
- The conference has two tracks: one on general theory and the other on programming languages.
- The goal of the conference is to introduce students to various topics in computer science and make them accessible.
- The conference allows attendees to switch between tracks based on their interests.
- The conference will take place on 23rd February at the end of Flexible Learning Week.
- Tickets for the conference cost £5, but a discount code is available for £1 off.
- The ticket price includes catering for the whole day and access to all talks and branded items.

## Introduction to Stack and Heap

- Stack and heap are two memory structures used by Java to store data.
- Stack is a temporary memory structure where local variables are stored.
- Heap is a memory structure where objects are stored.

- Objects are passed by reference, meaning that when an object is passed to a function, it is not copied, but a reference to its memory location is passed.
- The "new" keyword is used to allocate memory for objects on the heap.
- The stack is limited in size and can cause a stack overflow if it is filled up.
- Java automatically manages the heap memory and performs garbage collection to free up memory that is no longer in use.

## Side Effects and Immutability

- Side effects occur when changes made to an object in one part of the code affect its behavior in another part of the code.
- Immutable objects are objects that cannot be changed once they are created.
- Immutable objects can help prevent side effects and make code more predictable.
- Java provides ways to mark objects as immutable to prevent accidental modifications.

## Examples and Applications

### Example 1: Circle Class

```java
public class Circle {
public double radius;
public Circle(double radius) {
this.radius = radius;
}
public double getArea() {
return Math.PI * radius * radius;
}
public void enlarge(double factor) {
radius *= factor;
```

}

public boolean equals(Circle other) {

return this.radius == other.radius;

}

}

- The Circle class represents a circle with a given radius.
- The class has a constructor, a method to calculate the area, a method to enlarge the circle, and an equals method to compare circles based on their radius.

### Example 2: Stack and Heap Memory

```java
int jackMoney = 100;

int jackTarget = 500;

double weeks = 2.5;

int[] numbers = {1, 2, 3, 4, 5};

String name = "John";
```

- In this example, the variables `jackMoney`, `jackTarget`, and `weeks` are stored on the stack.
- The array `numbers` and the string `name` are stored on the heap.

### Example 3: Shallow Copy vs Deep Copy

```java
Circle[] someCircles = new Circle[5];

Circle[] shallowCopy = someCircles;

Circle[] deepCopy = new Circle[5];

for (int i = 0; i < someCircles.length; i++) {

deepCopy[i] = new Circle(someCircles[i].radius);

}
```

```

- In this example, `shallowCopy` is a shallow copy of `someCircles`, meaning that both arrays reference the same objects in memory.
- `deepCopy` is a deep copy of `someCircles`, meaning that a new array is created with new objects that have the same values as the original array.

## Best Practices and Warnings

- Be aware of side effects when working with objects and ensure that modifications to objects do not have unintended consequences.
- Use immutability when possible to prevent accidental modifications and make code more predictable.
- When copying objects, be careful to create deep copies if necessary to avoid unintended sharing of references.
- Understand the difference between stack and heap memory and their limitations.
- Be mindful of memory usage and use garbage collection to free up memory that is no longer in use.

## Comparisons and Alternatives

- The stack and heap are memory structures used by Java to store data, and each has its own advantages and limitations.
- The stack is limited in size and is used for storing local variables, while the heap is used for storing objects and is managed by the Java Virtual Machine.
- Other programming languages may use different memory structures or have different memory management strategies.
- It is important to understand the memory structures and management techniques used in the programming language you are working with to write efficient and reliable code.
```