Tutorial 2

Question 1

1. **Analyse the program below and discuss the issue. How can this program be improved?**

```java
/*
 * @author Aarthi N
 * An example to understand that Java Strings are immutable
 */
public class StringExample {

	public static void main(String[] args) {
		String message = "It is pouring like cats and dogs outside";
		message = message + "and the lightining struck [";
		for (int i=0; i<=11; i++){
			message = message + i + ",";
		}
		message = message + "]";
		System.out.println(message);
	}
}
```

Java Strings

- From Java Documentation:

  “Strings are constant; their values cannot be changed after they are created.”

- What does this mean?
- How can string concatenation (`str3 = str1 + str2`) be achieved?

1. **Analyse the program below and discuss the issue. How can this program be improved?**

```java
/*
 * @author Aarthi N
 * An example to understand that Java Strings are immutable
 */
public class StringExample {

	public static void main(String[] args) {
		String message = "It is pouring like cats and dogs outside";
		message = message + "and the lightining struck [";
		for (int i=0; i<=11; i++){
			message = message + i + ",";
		}
		message = message + "]";
		System.out.println(message);
	}
}
```

Solution

- Mutable String Classes
- **StringBuilder**, StringBuffer

Question 2

Create an Employee class which has private fields for an Employee’s name and salary

Create a Manager class which is a subtype of Employee

What is the difference between super and this ?

What does super() and this() refer to?

What is method overriding?

Aside: The class **Object**

Object Functionality

- Are there any operations all Objects should support?
- Equality Check
- Get a String representation of the Object (for Logging, Debugging, etc)
- others..

- How do we make sure all Classes written support this functionality?

class Object {}

- All classes inherit from this
- Defines common methods
- **equals()** (eg: **obj1.equals(obj2)**)
- **toString()**
- **getClass()**

- more..check documentation
- How can Java know how to represent your custom class as a String?

What is method overriding?

Can you override a static method?

If a Manager object is declared as an Employee, which methods of Manager can you call on this object?

Encapsulation

Encapsulation

- Fundamental OOP concept
- Hide / restrict direct access to fields (**obj.field**) and provide methods to do so instead
- Why is this useful / necessary?

- Example: Bank Account class - control ways of changing balance field

Implementation

- Use visibility keyword for fields - **private** fields cannot be directly accessed
- Control access through availability of public methods - called **getters** and **setters**
- **Getters** retrieve value, **setters** set/modify value

- Example: Don’t provide setter for Balance in BankAccount class -> code cannot change the balance at will

Inheritance

Modelling the Real World

- Classes/Objects can be useful in modelling real world entities
- How can we model the relationships between these entities?

Inheritance

- Provides a way of “building up” classes to create new classes
- Inherited classes have common features (fields)
- They share the same methods as their parents (implementation may differ)

- **They can add new fields / methods**

Subtyping

- In Java, inheritance provides a way of subtyping
- Example: base class **Shape**
- Child Classes Rectangle, Triangle, etc..

- A Rectangle is a Shape, but Triangles are not Rectangles