Tutor Marked Exercises 3, Questions, Design Document, Testing Plan

Unit 6,7 Learning Objective Questions

Purpose of a `Class` Object (TIJ Page 556)

In Java, a `Class` object is a special object that represents a class or interface at runtime. It is part of the **Reflection API** and allows you to examine or manipulate the structure and behavior of a class during runtime.

Uses of a `Class` Object:

1. **Dynamic Loading**: Load a class at runtime using `Class.someNaem("className")`.
2. **Retrieve**: Retrieve metadata like methods, fields, constructors, and annotations.
3. **Object Creation**: Create new instances of a class using `new ClassName();`.
4. **Type Checking**: Check if an object is an instance of a specific class.

Using `enum` Inside a `switch` Statement (TIJ Pages 1016–1017)

Enums are commonly used in `switch` statements to make the code more readable and type-safe.

Example:

```
enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, S
UNDAY;
}

public class Main {
    public static void main(String[] args) {
        Day today = Day.WEDNESDAY;

        switch (today) {
            case MONDAY:
                System.out.println("Start of the work wee
k.");
                break;
            case FRIDAY:
                System.out.println("End of the work week.");
                break;
            case SUNDAY:
                System.out.println("Weekend is almost ove
r.");
                break;
            default:
                System.out.println("Midweek day.");
        }
    }
}
```

Output:

```
Midweek day.
```

Difference Between TCP and UDP

Feature	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Connection	Connection-oriented: Establishes a connection before data transfer.	Connectionless: No connection setup is required.
Reliability	Reliable: Ensures delivery, error checking, and retransmission.	Unreliable: No guarantee of delivery or error checking.
Speed	Slower due to reliability and connection overhead.	Faster due to minimal overhead.
Use Case	Used for applications requiring reliable communication (e.g., HTTP, FTP, email).	Used for time-sensitive applications where speed matters (e.g., video streaming, DNS, gaming).
Data Transfer	Data is transferred as a stream (segments).	Data is transferred as individual packets (datagrams).
Header Size	Larger header (20 bytes or more).	Smaller header (8 bytes).

Default Port Number in a URL

If the port number is not specified in a URL, the default port is determined based on the protocol:

1. **HTTP**: Port `80`
2. **HTTPS**: Port `443`

Example:

- URL: `http://example.com` → Defaults to `http://example.com:80`.
- URL: `https://example.com` → Defaults to `https://example.com:443`.

TME 3: Test Plan and Design Document

Course: Computer Science 308 – Java for Programmers

Assignment: TME 3

Author: Boris Bojanov

Student ID: 3608550

Date: Jan 28 2025

---

1. Overview

This TME involves implementing and extending the `GreenhouseControls` design with additional functionality, including new events, exception handling, serialization, and restoration capabilities. The assignment progresses in four steps, with each step building upon the previous.

---

2. Test Plan

2.1 Objectives

- Verify correctness of newly implemented functionality in each step.
- Ensure the system behaves as expected under normal, abnormal, and edge-case scenarios.
- Test exception handling, logging, serialization, and restoration processes.

2.2 Compile & Run Instructions

1. **Compilation:**
   - `javac -d bin src/tme3/*.java`
2. **Execution:**
   - Step 2: `java GreenhouseControls -f examples1.txt`
   - Step 3: `java GreenhouseControls -f examples3.txt`
   - Step 4: `java GreenhouseControls -d dump.out`

2.3 Test Cases

Step 2: Adding Functionality to Existing Design

Test Case 1: FansOn and FansOff Events

- **Purpose:** Verify that the `fans` variable is toggled correctly.
- **Input:** Trigger `FansOn` and `FansOff` events.
- **Expected Result:**
  - `FansOn` sets `fans = true`.
  - `FansOff` sets `fans = false`.
- **Actual Result:** [Record Actual Result]

Test Case 2: Modified Bell Event

- **Purpose:** Ensure multiple bell events execute with a 2000 ms delay.
- **Input:** Use `examples1.txt` with multiple bell events.
- **Expected Result:**
  - Bells ring the specified number of times with a 2000 ms interval.
  - Other events execute between bell rings as expected.
- **Actual Result:** [Record Actual Result]

Step 3: Simulating Problems

Test Case 3: WindowMalfunction Event

- **Purpose:** Test the functionality of `WindowMalfunction`.
- **Input:** Trigger a `WindowMalfunction` event.
- **Expected Result:**
  - Sets `windowok = false`.
  - Throws a `ControllerException` with the appropriate error message.
  - Logs error details to `error.log` and serializes the state to `dump.out`.
- **Actual Result:** [Record Actual Result]

Test Case 4: PowerOut Event

- **Purpose:** Verify the functionality of `PowerOut`.

- **Input:** Trigger a `PowerOut` event.
- **Expected Result:**
  - Sets `poweron = false`.
  - Throws a `ControllerException` with the appropriate error message.
  - Logs error details to `error.log` and serializes the state to `dump.out`.
- **Actual Result:** [Record Actual Result]

Test Case 5: Emergency Shutdown

- **Purpose:** Ensure `shutdown` method executes correctly after exceptions.
- **Input:** Trigger either `WindowMalfunction` or `PowerOut`.
- **Expected Result:**
  - Logs error details.
  - Serializes the state and terminates execution.
- **Actual Result:** [Record Actual Result]

Step 4: System Restore

Test Case 6: Restoring System State

- **Purpose:** Test restoration of the serialized `GreenhouseControls` object.
- **Input:** Run `Restore` with `dump.out`.
- **Expected Result:**
  - Restores the state of the system.
  - Fixes the issue using the appropriate `Fixable` implementation.
  - Logs restoration details to `fix.log` and resumes execution.
- **Actual Result:** [Record Actual Result]

Test Case 7: Fixable Interface

- **Purpose:** Verify the functionality of `Fixable` implementations (`PowerOn`, `FixWindow`).

- **Input:** Invoke `getFixable(int errorcode)`.
- **Expected Result:**
  - Returns the correct `Fixable` implementation.
  - Fixes the issue and resets error code to zero.
- **Actual Result:** [Record Actual Result]

---

3. Design Document

3.1 Classes and Methods

3.1.1 GreenhouseControls

- **Fields:**
  - `boolean fans`
  - `boolean windowok`
  - `boolean poweron`
  - `int errorcode`
- **Methods:**
  - `void action()` (override in events)
  - `void shutdown()`
  - `Fixable getFixable(int errorcode)`
  - `int getError()`

3.1.2 Events

- **FansOn, FansOff:** Toggle `fans` variable.
- **WindowMalfunction:** Set `windowok = false` and throw `ControllerException`.
- **PowerOut:** Set `poweron = false` and throw `ControllerException`.
- **Bell:** Execute multiple times with a delay.

3.1.3 Fixable Interface

- **Methods:**
  - `void fix()`
  - `void log()`

3.1.4 Restore

- **Methods:**
  - Reads `dump.out` to restore `GreenhouseControls`.
  - Calls `getFixable` and resumes execution.

3.2 Algorithms

Bell Event Execution

1. Parse the number of rings.
2. Schedule `Event` objects with a 2000 ms delay.
3. Allow interleaving with other events.

Exception Handling

1. Catch `ControllerException`.
2. Call `shutdown()` to log and serialize state.
3. Display error details in the console.

System Restoration

1. Deserialize `GreenhouseControls` from `dump.out`.
2. Use `Fixable` to restore system state.
3. Resume execution from the next scheduled event.

---

4. Summary

- **Submission Requirements:**

    - All source files.
    - This test plan.
    - Answers to Unit 6 and 7 review questions.
- **Evaluation Criteria:**
    - Correctness of implementation (95%).
    - Completeness of review question answers (5%).

End of Document

NOTE:

The following link is not working properly!!!

“Please see testplan.html for a sample test plan.” (https://scis.lms.athabascau.ca/file.php/422/tme_files/guidelines.htm)

Results in a black page. DegreeWorks does not present me with a way to find the template of a test plan. I have sent Emails requesting an example test plan.