Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
по дисциплине «Надёжность программного обеспечения»
Тема: ***«Реализация и анализ отказоустойчивости»***

Выполнил:
Студент 3-го курса,
ФЭИС Группы ПО-11
Лесько М.И.
Проверил:
Козик И. Д.

Брест 2025

Цель работы: Разработка отказоустойчивой системы и анализ ее поведения при сбоях.

Задание: Разработайте программу, которая имитирует отказоустойчивую систему (например, резервирование данных, переключение на backup-сервер и т.д.). Реализуйте механизмы восстановления после сбоев (например, перезапуск сервиса, использование резервных данных). Проведите тестирование, искусственно вызывая сбои (например, отключение сети, сбои в работе оборудования). Оцените время восстановления и предложите улучшения.

Вариант 12: Резервирование сетевых соединений: Создайте программу, которая переключается на резервный сервер при недоступности основного.

Ход работы

Код:

Main.java

```
public class Main {
   public static void main(String[] args) {
      try {
         System.out.println("ЗАПУСК СИСТЕМЫ
ОТКАЗОУСТОЙЧИВОСТИ СЕРВЕРОВ");
         System.out.println("==============================
=============");

         // Создание серверов
         NetworkServer primaryServer = new
NetworkServer("Основной сервер");
         NetworkServer backupServer = new
NetworkServer("Резервный сервер");

         // Создание отказоустойчивой системы
         FaultTolerantSystem system = new
FaultTolerantSystem(primaryServer, backupServer);

         // Начальное подключение
         System.out.println("\nЭТАП 1: Начальное подключение
к системе");
         system.connect();

         // Симуляция сбоя основного сервера
         System.out.println("\nЭТАП 2: Тестирование
отказоустойчивости");
         System.out.println("Симулируем сбой основного
сервера...");
         primaryServer.simulateFailure();

         // Попытка подключения (должен сработать
переход на резервный сервер)
         System.out.println("\nЭТАП 3: Автоматическое
переключение");
         System.out.println("Проверяем работу механизма
переключения...");
         system.connect();

         // Симуляция сбоя резервного сервера
         System.out.println("\nЭТАП 4: Тестирование
критического сбоя");
         System.out.println("Симулируем сбой резервного
сервера...");
         backupServer.simulateFailure();

         // Попытка подключения (должна завершиться
ошибкой)
         System.out.println("\nЭТАП 5: Проверка обработки
полного отказа");
         System.out.println("Пытаемся подключиться к
системе...");
         try {
            system.connect();
         } catch (Exception e) {
```

```
            System.out.println("ИНФОРМАЦИЯ: " +
e.getMessage());
        }

        // Восстановление основного сервера
        System.out.println("\nЭТАП 6: Восстановление
системы");
        System.out.println("Восстанавливаем работу
основного сервера...");
        primaryServer.recover();

        // Попытка подключения (должна быть успешной)
        System.out.println("\nЭТАП 7: Проверка
восстановления");
        System.out.println("Проверяем работу
восстановленной системы...");
        system.connect();

        // Вывод статистики
        System.out.println("\nИТОГОВАЯ СТАТИСТИКА
СИСТЕМЫ");
        System.out.println("----------------------------------------");
        System.out.println("Количество переключений на
резервный сервер: " + system.getFailoverCount());
        System.out.println("Время последнего переключения:
" + system.getLastFailoverTime() + " миллисекунд");
        System.out.println("Текущий активный сервер: " +
system.getCurrentServer().getServerName());
        System.out.println("\nТестирование системы
завершено успешно");

      } catch (Exception e) {
        System.out.println("\nКРИТИЧЕСКАЯ ОШИБКА: " +
e.getMessage());
      }
   }
}
```

FaultTolerantSystem.java

```
public class FaultTolerantSystem {
   private final Server primaryServer;
   private final Server backupServer;
   private Server currentServer;
   private long lastFailoverTime;
   private int failoverCount;
   private boolean isHealthCheckActive;
   private Thread healthCheckThread;
   private static final long HEALTH_CHECK_INTERVAL = 5000; // 5 секунд

   public FaultTolerantSystem(Server primaryServer, Server backupServer) {
      this.primaryServer = primaryServer;
      this.backupServer = backupServer;
      this.currentServer = primaryServer;
      this.failoverCount = 0;
      this.isHealthCheckActive = false;
      startHealthCheck();
   }

   private void startHealthCheck() {
      isHealthCheckActive = true;
      healthCheckThread = new Thread(() -> {
         while (isHealthCheckActive) {
            try {
               checkServersHealth();
               Thread.sleep(HEALTH_CHECK_INTERVAL);
            } catch (InterruptedException e) {
               System.out.println("ИНФОРМАЦИЯ: Проверка состояния серверов прервана");
            }
         }
      });
      healthCheckThread.start();
   }

   private void checkServersHealth() {
```

```
    if (currentServer == primaryServer && !primaryServer.isAvailable() && backupServer.isAvailable()) {
        try {
            System.out.println("СИСТЕМА: Обнаружена неисправность основного сервера, инициируем превентивное переключение");
            performFailover();
        } catch (Exception e) {
            System.out.println("ОШИБКА: Не удалось выполнить превентивное переключение: " + e.getMessage());
        }
    }
}

public void connect() throws Exception {
    try {
        if (currentServer.isAvailable()) {
            currentServer.connect();
        } else {
            performFailover();
        }
    } catch (Exception e) {
        performFailover();
    }
}

private void performFailover() throws Exception {
    System.out.println("\nСИСТЕМА: Начинается процесс переключения на резервный сервер...");
    long startTime = System.currentTimeMillis();

    if (currentServer == primaryServer) {
        currentServer.disconnect();
        currentServer = backupServer;
        System.out.println("СИСТЕМА: Производится переключение с основного на резервный сервер");
    } else {
        currentServer.disconnect();
        currentServer = primaryServer;
        System.out.println("СИСТЕМА: Производится переключение с резервного на основной сервер");
    }

    if (!currentServer.isAvailable()) {
        throw new Exception("КРИТИЧЕСКАЯ ОШИБКА: Все серверы в системе недоступны. Требуется вмешательство администратора");
    }

    currentServer.connect();
    lastFailoverTime = System.currentTimeMillis() - startTime;
    failoverCount++;
    System.out.println("СИСТЕМА: Переключение успешно завершено за " + lastFailoverTime + " миллисекунд");

    // Анализ времени переключения
    analyzeFailoverTime(lastFailoverTime);
}

private void analyzeFailoverTime(long time) {
    if (time > 1000) { // Если переключение заняло больше 1 секунды
        System.out.println("ПРЕДУПРЕЖДЕНИЕ: Время переключения превышает оптимальное значение");
        System.out.println("РЕКОМЕНДАЦИЯ: Проверьте нагрузку на серверы и сетевую инфраструктуру");
    }
}

public void disconnect() {
    isHealthCheckActive = false;
    if (healthCheckThread != null) {
        healthCheckThread.interrupt();
    }
    currentServer.disconnect();
}

public long getLastFailoverTime() {
    return lastFailoverTime;
}

public int getFailoverCount() {
    return failoverCount;
}
```

```
    public Server getCurrentServer() {
        return currentServer;
    }
}
NetworkServer.java
public class NetworkServer implements Server {
    private final String serverName;
    private boolean isConnected;
    private boolean isSimulatedFailure;
    private int failureCount;
    private static final int MAX_FAILURES = 3;
    private long lastFailureTime;
    private static final long RECOVERY_TIMEOUT = 30000; // 30 секунд

    public NetworkServer(String serverName) {
        this.serverName = serverName;
        this.isConnected = false;
        this.isSimulatedFailure = false;
        this.failureCount = 0;
        this.lastFailureTime = 0;
    }

    @Override
    public boolean isAvailable() {
        // Проверяем, не истек ли таймаут восстановления
        if (isSimulatedFailure && System.currentTimeMillis() - lastFailureTime > RECOVERY_TIMEOUT) {
            attemptAutoRecovery();
        }
        return !isSimulatedFailure;
    }

    private void attemptAutoRecovery() {
        if (failureCount < MAX_FAILURES) {
            recover();
            System.out.println("СИСТЕМА: Автоматическое восстановление " + serverName + " выполнено успешно");
        } else {
            System.out.println("ПРЕДУПРЕЖДЕНИЕ: " + serverName + " достиг максимального количества сбоев. Требуется ручное
вмешательство");
        }
    }

    @Override
    public String getServerName() {
        return serverName;
    }

    @Override
    public void connect() throws Exception {
        if (isSimulatedFailure) {
            throw new Exception("ВНИМАНИЕ: " + serverName + " не отвечает. Сервер временно недоступен.");
        }
        isConnected = true;
        System.out.println("УСПЕШНО: Соединение с " + serverName + " установлено и активно");
    }

    @Override
    public void disconnect() {
        isConnected = false;
        System.out.println("ИНФОРМАЦИЯ: Соединение с " + serverName + " было безопасно закрыто");
    }

    public void simulateFailure() {
        isSimulatedFailure = true;
        failureCount++;
        lastFailureTime = System.currentTimeMillis();
        System.out.println("ОШИБКА: На " + serverName + " обнаружен критический сбой. Сервер переведен в нерабочее состояние");
        System.out.println("ИНФОРМАЦИЯ: Количество сбоев: " + failureCount + " из " + MAX_FAILURES);
    }

    public void recover() {
        isSimulatedFailure = false;
        System.out.println("ВОССТАНОВЛЕНИЕ: " + serverName + " успешно восстановлен и готов к работе");
```

```
    }

    public int getFailureCount() {
        return failureCount;
    }
}
```

Server.java

```
public interface Server {
    boolean isAvailable();
    String getServerName();
    void connect() throws Exception;
    void disconnect();
}
```

Исход работы программы:

```
ЗАПУСК СИСТЕМЫ ОТКАЗОУСТОЙЧИВОСТИ СЕРВЕРОВ
==========================================

ЭТАП 1: Начальное подключение к системе
УСПЕШНО: Соединение с Основной сервер установлено и активно

ЭТАП 2: Тестирование отказоустойчивости
Симулируем сбой основного сервера...
ОШИБКА: На Основной сервер обнаружен критический сбой. Сервер переведен в нерабочее состояние

ЭТАП 3: Автоматическое переключение
Проверяем работу механизма переключения...

СИСТЕМА: Начинается процесс переключения на резервный сервер...
ИНФОРМАЦИЯ: Соединение с Основной сервер было безопасно закрыто
СИСТЕМА: Производится переключение с основного на резервный сервер
УСПЕШНО: Соединение с Резервный сервер установлено и активно
СИСТЕМА: Переключение успешно завершено за 0 миллисекунд

ЭТАП 4: Тестирование критического сбоя
Симулируем сбой резервного сервера...
ОШИБКА: На Резервный сервер обнаружен критический сбой. Сервер переведен в нерабочее состояние

ЭТАП 5: Проверка обработки полного отказа
Пытаемся подключиться к системе...

СИСТЕМА: Начинается процесс переключения на резервный сервер...
ИНФОРМАЦИЯ: Соединение с Резервный сервер было безопасно закрыто
СИСТЕМА: Производится переключение с резервного на основной сервер

СИСТЕМА: Начинается процесс переключения на резервный сервер...
ИНФОРМАЦИЯ: Соединение с Основной сервер было безопасно закрыто
СИСТЕМА: Производится переключение с основного на резервный сервер
ИНФОРМАЦИЯ: КРИТИЧЕСКАЯ ОШИБКА: Все серверы в системе недоступны. Требуется вмешательство администратора
```

```
ЭТАП 6: Восстановление системы
Восстанавливаем работу основного сервера...
ВОССТАНОВЛЕНИЕ: Основной сервер успешно восстановлен и готов к работе

ЭТАП 7: Проверка восстановления
Проверяем работу восстановленной системы...

СИСТЕМА: Начинается процесс переключения на резервный сервер...
ИНФОРМАЦИЯ: Соединение с Резервный сервер было безопасно закрыто
СИСТЕМА: Производится переключение с резервного на основной сервер
УСПЕШНО: Соединение с Основной сервер установлено и активно
СИСТЕМА: Переключение успешно завершено за 0 миллисекунд

ИТОГОВАЯ СТАТИСТИКА СИСТЕМЫ
----------------------------------------
Количество переключений на резервный сервер: 2
Время последнего переключения: 0 миллисекунд
Текущий активный сервер: Основной сервер

Тестирование системы завершено успешно
```