

Design Patterns :-

Builder Design Pattern:

I am using lombok library to achieve builder pattern in my application.

```
@Data
@Builder
public class TokenDto {

    private String token;
    private Date expiration;
}
```

```
@Override
public TokenDto generateToken(String subject) {
    Map<String, Object> claims = new HashMap<>();
    Date expiration = new Date(System.currentTimeMillis() + jwtConfig.getExpiration());
    String token = Jwts.builder()
        .setClaims(claims)
        .setSubject(subject)
        .setIssuedAt(new Date(System.currentTimeMillis()))
        .setExpiration(expiration)
        .signWith(SignatureAlgorithm.HS512, getSecretKey(jwtConfig.getSecret()))
        .compact();

    return TokenDto.builder()
        .token(token)
        .expiration(expiration)
        .build();
}
```

Singleton Design pattern:

I am using spring boot for my application, so i am using spring annotation to create Singleton object of my class. For example `@Service`, `@RestController` and so on.

```
no usages  mannu717
@Service
public class JwtServiceImpl implements JwtService {

    4 usages
    private final JwtConfig jwtConfig;

    no usages  mannu717
    public JwtServiceImpl(JwtConfig jwtConfig) { this.jwtConfig = jwtConfig; }
```

Proxy Design Pattern:

In my application to get apartment details i am using proxy design pattern, which will first check if data available in local cache otherwise call actual service to pull data from DB.

```
2 usages  mannu717
@Override
@CachePut(value = "apartment_cache", key = "#result.id", condition = "#result != null"), cacheable = { @Cacheable(value = "apartment_cache", key = "#result.id", condition = "#result != null"), })
public ApartmentResponseDto createApartment(ApartmentRequestDto apartmentDto) {
    return apartmentService.createApartment(apartmentDto);
}

2 usages  mannu717
@Override
@CachePut(value = "apartment_cache", key = "#apartmentId")
public ApartmentResponseDto updateApartment(String apartmentId, ApartmentRequestDto updatedApartmentRequestDto) {
    return apartmentService.updateApartment(apartmentId, updatedApartmentRequestDto);
}

2 usages  mannu717
@Override
@CacheEvict(value = "apartment_cache", key = "#apartmentId")
public void deleteApartment(String apartmentId) { apartmentService.deleteApartment(apartmentId); }

2 usages  mannu717
@Override
@Cacheable(value = "apartment_cache", key = "#apartmentId")
public ApartmentResponseDto getApartmentDetails(String apartmentId) {
    return apartmentService.getApartmentDetails(apartmentId);
}
```

Strategy Design Pattern:

I am using strategy design pattern in my application while i am sending login OTP details to user via email and SMS.

```
2 usages
private final CommunicationService emailCommunicationService;
2 usages
private final CommunicationService smsCommunicationService;

no usages  ⬆️ mannu717
public AuthenticationServiceImpl(OTPService otpService, OTPMapper otpMapper, UserService userService, JwtService jwtService,
    @Qualifier("emailCommunicationService") CommunicationService emailCommunicationService,
    @Qualifier("smsCommunicationService") CommunicationService smsCommunicationService) {
    this.otpService = otpService;
    this.otpMapper = otpMapper;
    this.userService = userService;
    this.jwtService = jwtService;
    this.emailCommunicationService = emailCommunicationService;
    this.smsCommunicationService = smsCommunicationService;
}
```

```
@Override
public LoginResponse userLogin(LoginRequest request) throws OTPException, LoginException {
    String email = request.getEmail();
    userService.throwExceptionIfNotExist(email);
    User user = userService.findByEmail(email);

    String otp = request.getOtp();
    if (!otpService.validateOtp(email, otp)) {
        throw new LoginException("Invalid OTP");
    }

    TokenDto tokenDto = jwtService.generateToken(otp);

    //Send otp to user via SMS and EMail
    emailCommunicationService.sendMessage(user.getEmail(), otp);
    smsCommunicationService.sendMessage(user.getMobile(), otp);

    return LoginResponse.builder()
        .token(tokenDto.getToken())
        .expirationTime(tokenDto.getExpiration().toInstant())
        .build();
}
```

SOLID Principles: -

Single Responsibility Principle:

I adopt microservice architecture for my application so that i can divide responsibilities between services. Also all my classes in my project are following Single responsibility principle.

Dependency Inversion Principle:-

Spring Boot achieves the Dependency Inversion Principle (DIP) by supporting dependency injection through annotations like `@Autowired`, managing bean lifecycles with an Inversion of Control (IoC) container, enabling configuration of concrete implementations as beans, facilitating the use of profiles to switch between implementations, and encouraging constructor injection to enforce dependencies through abstractions, promoting modular and maintainable code adhering to DIP principles.

JVM Flags: -

-Xss128k – For setting up the stack size to 128kb.

-XX:+UseG1GC - To enable the G1 (Garbage-First) garbage collector in the JVM

-XX:MaxGCPauseMillis - Specifies the maximum pause time goal for the G1 garbage collector in milliseconds.

-XX:GCTimeRatio - Specifies the ratio of time spent in garbage collection versus application-level processing.

```
FROM openjdk:17
VOLUME /tmp
EXPOSE 8080
ARG JAR_FILE=target/apartment-management-service-1.0.0.jar
ADD ${JAR_FILE} app.jar
ENV JAVA_OPTS="-Xss512k -XX:+UseG1GC -XX:GCTimeRatio=4 -XX:MaxGCPauseMillis=200"

# Copy your shell script into the image
COPY start.sh /start.sh

# Make the script executable (if needed)
RUN chmod +x /start.sh

CMD ["/start.sh"]
```

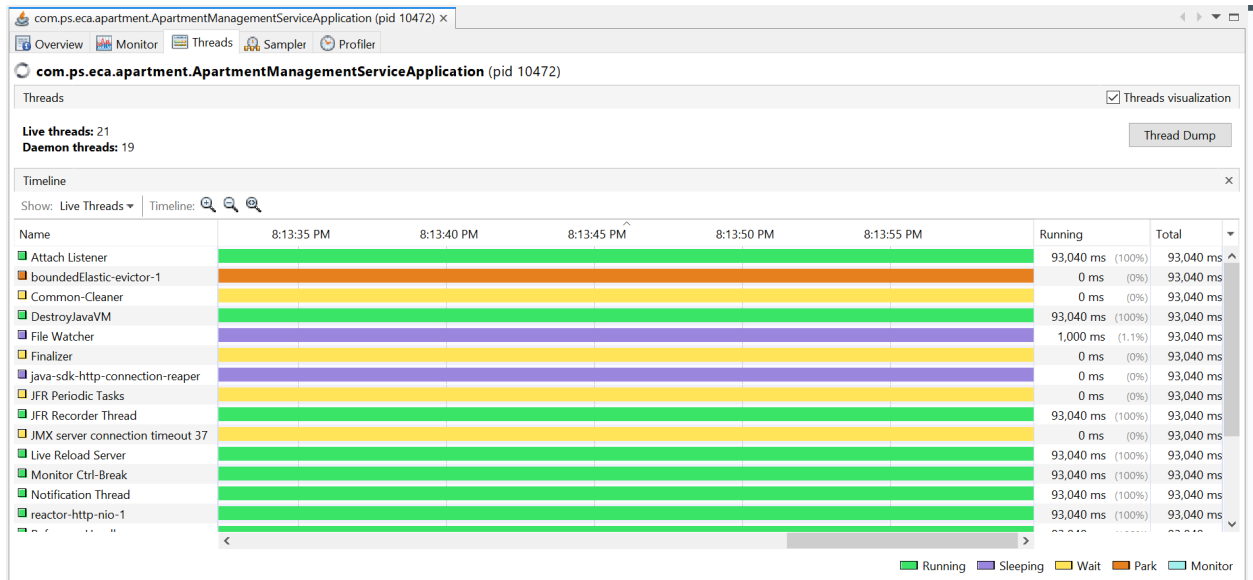
```
#!/bin/sh
```

```
# Start the Java application with JVM options
```

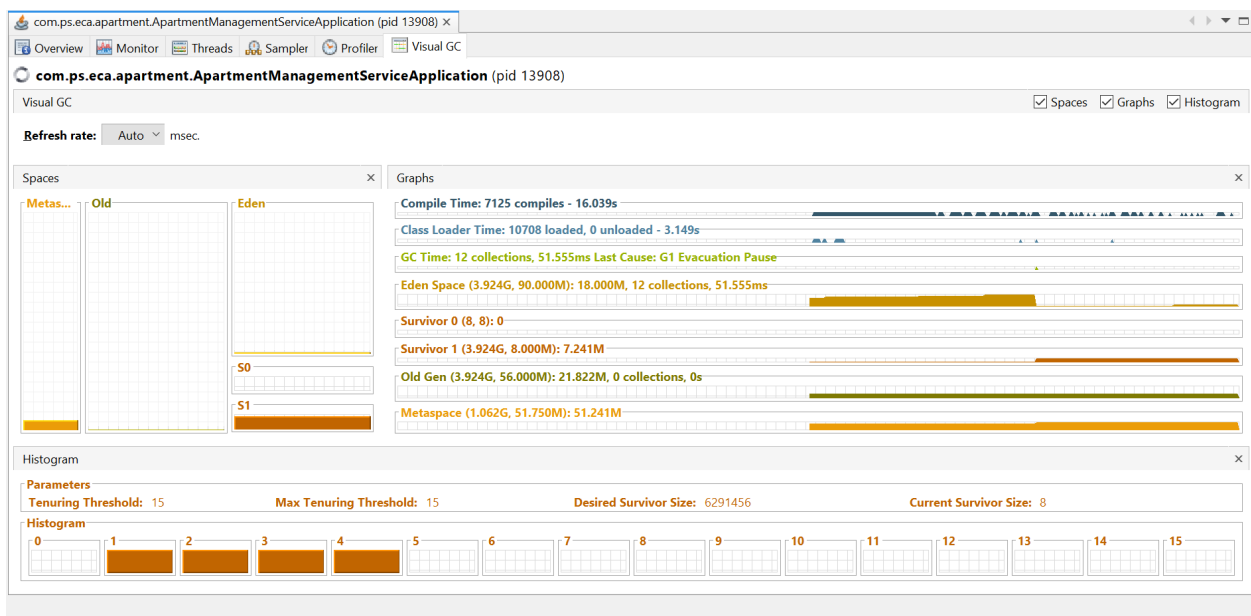
```
exec java $JAVA_OPTS -jar /app.jar
```

Visual VM Monitoring: -

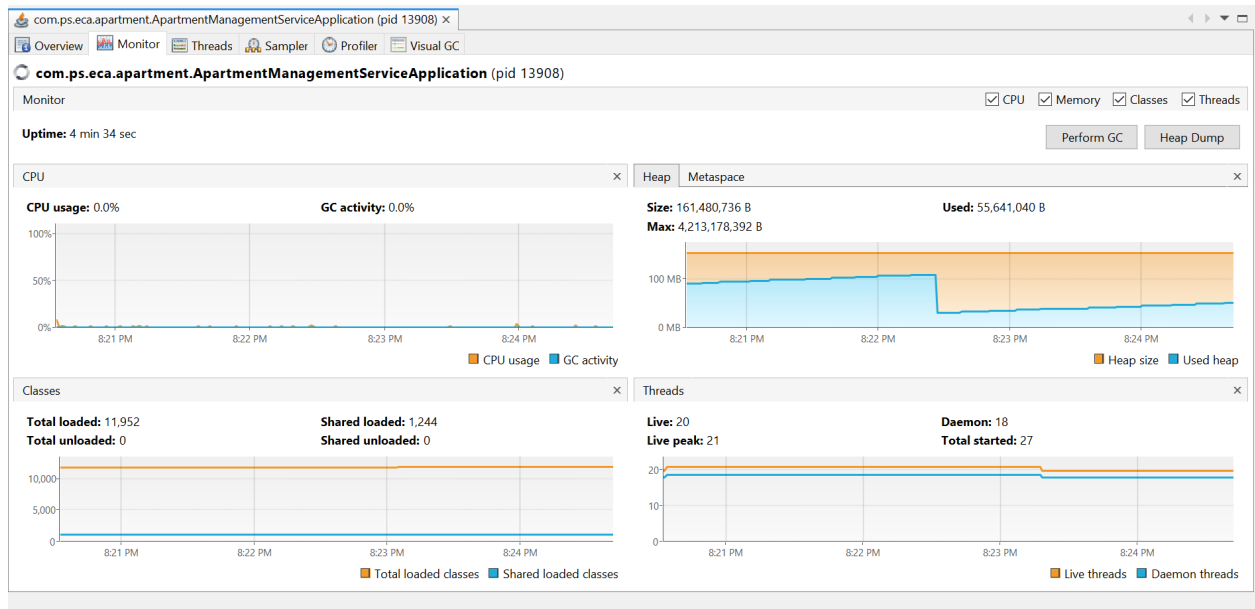
Threads :-



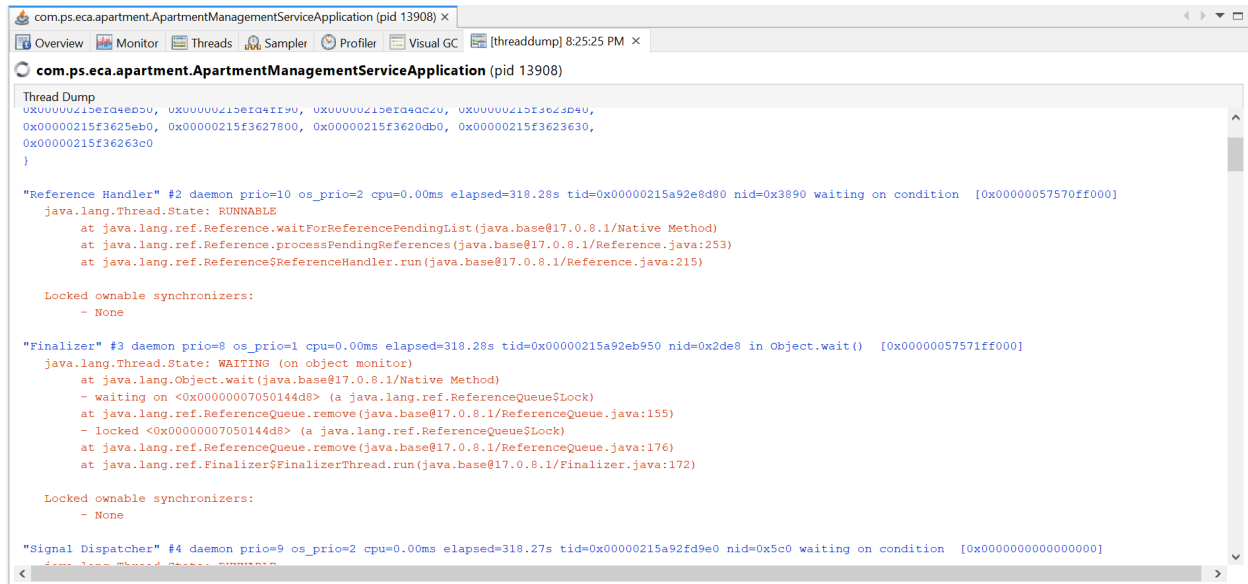
Memory:



Monitor:



Threaddump:



The screenshot shows a Java Threaddump window for the application 'com.ps.eca.apartment.ApartmentManagementServiceApplication (pid 13908)'. The window includes tabs for Overview, Monitor, Threads, Sampler, Profiler, and Visual GC. The 'Threads' tab is active, displaying a list of threads. The first thread is 'Reference Handler' #2, which is a daemon thread with priority 10, currently in a 'waiting on condition' state. The second thread is 'Finalizer' #3, also a daemon thread with priority 8, currently in a 'WAITING (on object monitor)' state. The third thread is 'Signal Dispatcher' #4, a daemon thread with priority 9, currently in a 'waiting on condition' state. The dump also shows the memory address of the thread and the state of the thread's lock and monitor.

```
com.ps.eca.apartment.ApartmentManagementServiceApplication (pid 13908) x
Overview Monitor Threads Sampler Profiler Visual GC [threaddump] 8:25:25 PM x

com.ps.eca.apartment.ApartmentManagementServiceApplication (pid 13908)

Thread Dump
0x00000215e9d4eb50, 0x00000215e9d4f190, 0x00000215e9d4ac20, 0x00000215f3623b40,
0x00000215f3625eb0, 0x00000215f3627800, 0x00000215f3620db0, 0x00000215f3623630,
0x00000215f36263c0
}

"Reference Handler" #2 daemon prio=10 os_prio=2 cpu=0.00ms elapsed=318.28s tid=0x00000215a92e8d80 nid=0x3890 waiting on condition [0x00000057570ff000]
  java.lang.Thread.State: RUNNABLE
    at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.8.1/Native Method)
    at java.lang.ref.Reference.processPendingReferences(java.base@17.0.8.1/Reference.java:253)
    at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.8.1/Reference.java:215)

  Locked ownable synchronizers:
    - None

"Finalizer" #3 daemon prio=8 os_prio=1 cpu=0.00ms elapsed=318.28s tid=0x00000215a92eb950 nid=0x2de8 in Object.wait() [0x00000057571ff000]
  java.lang.Thread.State: WAITING (on object monitor)
    at java.lang.Object.wait(java.base@17.0.8.1/Native Method)
    - waiting on <0x00000007050144d8> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(java.base@17.0.8.1/ReferenceQueue.java:155)
    - locked <0x00000007050144d8> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(java.base@17.0.8.1/ReferenceQueue.java:176)
    at java.lang.ref.Finalizer$FinalizerThread.run(java.base@17.0.8.1/Finalizer.java:172)

  Locked ownable synchronizers:
    - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=2 cpu=0.00ms elapsed=318.27s tid=0x00000215a92fd9e0 nid=0x5c0 waiting on condition [0x0000000000000000]
  java.lang.Thread.State: RUNNABLE
```

Multithreading: -

I have done some domain exercise for multithreading use case.

https://tools.publicis.sapient.com/bitbucket/projects/EOB/repos/manoj-bisht/browse/domain-exercise/domain-exercise/src/main/java/com/ps/eca/domain_exercise/multi_threading