Project:

healthcare-service

Scope:

Project 'healthcare-service'

Profile:

Default

Results:

Enabled coding rules: 30

- Checkstyle: 30

Problems found: 54

- Checkstyle: 54

Time statistics:

- Started: Mon Apr 08 16:57:30 EDT 2024
  - Initialization: 00:00:00.039
  - Checkstyle: 00:00:00.604
  - Gathering results: 00:00:00.439
- Finished: Mon Apr 08 16:57:31 EDT 2024

Detailed Results:

healthcare-service

- Efficiency
  - Hide Utility Class Constructor
    - Main
      - Utility classes should not have a public or default constructor. - line: 19
- Reliability
  - Design For Extension
    - BloodPressure
      - Class 'BloodPressure' looks like designed for extension (can be subclassed), but the method 'getHigh' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BloodPressure' final or making the method 'getHigh' static/final/abstract/empty, or adding allowed annotation for the method. - line: 18
      - Class 'BloodPressure' looks like designed for extension (can be subclassed), but the method 'getLow' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BloodPressure' final or making the method 'getLow' static/final/abstract/empty, or adding allowed annotation for the method. - line: 22
      - Class 'BloodPressure' looks like designed for extension (can be subclassed), but the method 'toString' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BloodPressure' final or making the method 'toString' static/final/abstract/empty, or adding allowed annotation for the method. - line: 26
      - Class 'BloodPressure' looks like designed for extension (can be subclassed), but the method 'equals' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BloodPressure' final or making the method 'equals' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34

- Class 'BloodPressure' looks like designed for extension (can be subclassed), but the method 'hashCode' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BloodPressure' final or making the method 'hashCode' static/final/abstract/empty, or adding allowed annotation for the method. - line: 43

- HealthInfo
  - Class 'HealthInfo' looks like designed for extension (can be subclassed), but the method 'getNormalTemperature' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'HealthInfo' final or making the method 'getNormalTemperature' static/final/abstract/empty, or adding allowed annotation for the method. - line: 20
  - Class 'HealthInfo' looks like designed for extension (can be subclassed), but the method 'getBloodPressure' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'HealthInfo' final or making the method 'getBloodPressure' static/final/abstract/empty, or adding allowed annotation for the method. - line: 24
  - Class 'HealthInfo' looks like designed for extension (can be subclassed), but the method 'toString' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'HealthInfo' final or making the method 'toString' static/final/abstract/empty, or adding allowed annotation for the method. - line: 28
  - Class 'HealthInfo' looks like designed for extension (can be subclassed), but the method 'equals' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'HealthInfo' final or making the method 'equals' static/final/abstract/empty, or adding allowed annotation for the method. - line: 36
  - Class 'HealthInfo' looks like designed for extension (can be subclassed), but the method 'hashCode' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'HealthInfo' final or making the method 'hashCode' static/final/abstract/empty, or adding allowed annotation for the method. - line: 45
- MedicalServiceImpl
  - Class 'MedicalServiceImpl' looks like designed for extension (can be subclassed), but the method 'checkBloodPressure' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'MedicalServiceImpl' final or making the method 'checkBloodPressure' static/final/abstract/empty, or adding allowed annotation for the method. - line: 19
  - Class 'MedicalServiceImpl' looks like designed for extension (can be subclassed), but the method 'checkTemperature' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'MedicalServiceImpl' final or making the method 'checkTemperature' static/final/abstract/empty, or adding allowed annotation for the method. - line: 28
- PatientInfo
  - Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'getId' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'getId' static/final/abstract/empty, or adding allowed annotation for the method. - line: 33
  - Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'getName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'getName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 37
  - Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'getSurname' does not have javadoc that explains how to do that safely. If class is not

designed for extension consider making the class 'PatientInfo' final or making the method 'getSurname' static/final/abstract/empty, or adding allowed annotation for the method. - line: 41
- Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'getBirthday' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'getBirthday' static/final/abstract/empty, or adding allowed annotation for the method. - line: 45
- Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'getHealthInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'getHealthInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 49
- Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'toString' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'toString' static/final/abstract/empty, or adding allowed annotation for the method. - line: 53
- Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'equals' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'equals' static/final/abstract/empty, or adding allowed annotation for the method. - line: 64
- Class 'PatientInfo' looks like designed for extension (can be subclassed), but the method 'hashCode' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfo' final or making the method 'hashCode' static/final/abstract/empty, or adding allowed annotation for the method. - line: 76

- PatientInfoFileRepository
  - Class 'PatientInfoFileRepository' looks like designed for extension (can be subclassed), but the method 'getById' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfoFileRepository' final or making the method 'getById' static/final/abstract/empty, or adding allowed annotation for the method. - line: 22
  - Class 'PatientInfoFileRepository' looks like designed for extension (can be subclassed), but the method 'add' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfoFileRepository' final or making the method 'add' static/final/abstract/empty, or adding allowed annotation for the method. - line: 37
  - Class 'PatientInfoFileRepository' looks like designed for extension (can be subclassed), but the method 'remove' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfoFileRepository' final or making the method 'remove' static/final/abstract/empty, or adding allowed annotation for the method. - line: 67
  - Class 'PatientInfoFileRepository' looks like designed for extension (can be subclassed), but the method 'update' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientInfoFileRepository' final or making the method 'update' static/final/abstract/empty, or adding allowed annotation for the method. - line: 72
- SendAlertServiceImpl
  - Class 'SendAlertServiceImpl' looks like designed for extension (can be subclassed), but the method 'send' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SendAlertServiceImpl' final or

making the method 'send' static/final/abstract/empty, or adding allowed annotation for the method. - line: 4

- Magic Number
  - Main
    - '26' is a magic number. - line: 30
    - '1980' is a magic number. - line: 30
    - '11' is a magic number. - line: 30
    - '80' is a magic number. - line: 31
    - '120' is a magic number. - line: 31
    - '16' is a magic number. - line: 35
    - '1982' is a magic number. - line: 35
    - '125' is a magic number. - line: 36
    - '78' is a magic number. - line: 36
    - '60' is a magic number. - line: 43
    - '120' is a magic number. - line: 43

- Usability
  - Hidden Field
    - BloodPressure
      - 'low' hides a field. - line: 13
      - 'high' hides a field. - line: 13
    - HealthInfo
      - 'normalTemperature' hides a field. - line: 14
      - 'bloodPressure' hides a field. - line: 15
    - MedicalServiceImpl
      - 'patientInfoRepository' hides a field. - line: 14
      - 'alertService' hides a field. - line: 14
    - PatientInfo
      - 'id' hides a field. - line: 16
      - 'name' hides a field. - line: 17
      - 'surname' hides a field. - line: 18
      - 'birthday' hides a field. - line: 19
      - 'healthInfo' hides a field. - line: 20
      - 'name' hides a field. - line: 29
      - 'healthInfo' hides a field. - line: 29
      - 'surname' hides a field. - line: 29
      - 'birthday' hides a field. - line: 29
    - PatientInfoFileRepository
      - 'mapper' hides a field. - line: 16
      - 'repoFile' hides a field. - line: 16