SAT: Software Quality and Code Style

Improve the code quality of the following programs:

1.

```
1  class Laboratory {
2
3      Microscope microscope;
4
5      Result analyze(Sample sample) {
6          if (microscope.isInorganic(sample) == true) {
7              return Result.INORGANIC;
8          } else {
9              return analyzeOrganic(sample);
10         }
11     }
12
13     private Result analyzeOrganic(Sample sample) {
14         if (microscope.isHumanoid(sample) == false) {
15             return Result.ALIEN;
16         } else {
17             return Result.HUMANOID;
18         }
19     }
20 }
```

```
class Laboratory {

    Microscope microscope;

    Result analyze(Sample sample) {
        if (microscope.isOrganic(sample)) {
            return analyzeOrganic(sample);
        } else {
            return Result.INORGANIC;
        }
    }

    private Result analyzeOrganic(Sample sample) {
        if (microscope.isHumanoid(sample)) {
            return Result.HUMANOID;
        } else {
            return Result.ALIEN;
        }
    }
}
```

2.

```
1   class Astronaut {
2
3       String name;
4       int missions;
5
6       boolean isValid() {
7           if (missions < 0 || name == null || name.trim().isEmpty()) {
8               return false;
9           } else {
10              return true;
11          }
12      }
13  }
```

```
class Astronaut {

    String name;
    int missions;

    boolean isValid() {
        return missions >= 0 && name != null && !name.trim().isEmpty();
    }
}

class Astronaut {
    String name = "";
    int missions = 1;

    boolean isValid() {
        boolean isValidMissions = missions >= 0;
        boolean isValidName = name != null && !name.trim().isEmpty();
        return isValidMissions && isValidName;
    }
}
```

3.

```
class SpaceShip {

    Crew crew;
    FuelTank fuelTank;
    Hull hull;
    Navigator navigator;
    OxygenTank oxygenTank;

    boolean willCrewSurvive() {
        return hull.holes == 0 &&
                fuelTank.fuel >= navigator.requiredFuelToEarth() &&
                oxygenTank.lastsFor(crew.size) > navigator.timeToEarth();
    }
}
```

```
class SpaceShip {

    Crew crew;
    FuelTank fuelTank;
    Hull hull;
    Navigator navigator;
    OxygenTank oxygenTank;

    boolean willCrewSurvive() {
        boolean hasEnoughResources = hasEnoughFuel() && hasEnoughOxygen();
        return hull.isIntact() && hasEnoughResources;
    }

    private boolean hasEnoughOxygen() {
        return oxygenTank.lastsFor(crew.size) > navigator.timeToEarth();
    }

    private boolean hasEnoughFuel() {
        return fuelTank.fuel >= navigator.requiredFuelToEarth();
    }
}
```

4.

```
class Logbook {

    void writeMessage(String message, Path location) throws IOException {
        if (Files.isDirectory(location)) {
            throw new IllegalArgumentException("The path is invalid!");
        }
        if (message.trim().equals("") || message == null) {
            throw new IllegalArgumentException("The message is invalid!");
        }
        String entry = LocalDate.now() + ": " + message;
        Files.write(location, Collections.singletonList(entry),
                StandardCharsets.UTF_8, StandardOpenOption.CREATE,
                StandardOpenOption.APPEND);
    }
}
```

```
class Logbook {

    void writeMessage(String message, Path location) throws IOException {
        if (message == null || message.trim().isEmpty()) {
            throw new IllegalArgumentException("The message is invalid!");
        }
        if (location == null || Files.isDirectory(location)) {
            throw new IllegalArgumentException("The path is invalid!");
        }

        String entry = LocalDate.now() + ": " + message;
        Files.write(location, Collections.singletonList(entry),
                StandardCharsets.UTF_8, StandardOpenOption.CREATE,
                StandardOpenOption.APPEND);
    }
}
```