```java
import lt.itakademija.exam.Issue;
import lt.itakademija.exam.Project;
import lt.itakademija.exam.ProjectManager;
import java.util.Collection;
import java.util.LinkedList;

public class ProjectManagerImp implements ProjectManager {

    private LinkedList<Project> projects = new LinkedList<>(); //sukuriam sarasa, nes juk projektus kazkur reikes deti
    LinkedList<Issue> issues = new LinkedList<>(); // sukuriam sarasa, nes juk ivykius kazkur reikes deti

    @Override
    public Project createProject(String id, String summary) { //pakeiciam s, s1 pagal doc
        if (id == null || summary == null) { // patikrinam ar projectas su duomenimis (id, summary)
            throw new NullPointerException();
        }
        if (id.equals("") || summary.equals("")) { //patikrinam ar stringas nera empty
            throw new IllegalArgumentException();
        }
        Project project = new Project(id, summary); //kadangi nera paduodamas metodui kaip argumentas, tai atsiranda poreikis projekta sukurti cia
        projects.add(project); //idedam projekta i sarasa
        return project;
    }

    @Override
    public Collection<Project> getProjects() { //nera poreikio iteruoti nes atiduodam visus projektus ( skliaustuose nera argumneto)
        return projects;
    }

    @Override
    public Project getProjectById(String id) {
```

```
32          for (Project project :projects) {
33              if (project.getId().equals(id)) {
34                  return project; //iskart returninu, nera poreikio deti i sarasa nes nepraso (atitinkamai ir saraso nekuriam)
35              }
36          }
37          return null;
38      }
39
40      @Override
41      public Issue createIssue(Project project, String summary) {
42          if (project == null || summary == null) {
43              throw new NullPointerException();
44          }
45          if (summary.equals("")) { //jeigu kaip argumentas priimamas string, tai be null reikia tikrinti ir empty
46              throw new IllegalArgumentException();
47          }
48          Issue issue = new Issue(project, summary); //argumentai pagal konstruktoriu
49          issues.add(issue);
50          return issue;
51      }
52
53      @Override
54      public Issue createIssue(String id, String summary) {
55          if (id == null || summary == null) {
56              throw new NullPointerException();
57          }
58          if (id.equals("") || summary.equals("")) { //jeigu kaip argumentas priimamas string, tai be null reikia tikrinti ir empty
59              throw new IllegalArgumentException();
60          }
61          Project project = getProjectById(id);
62          Issue issue = new Issue(project, summary);
```

```
63          //Issue issue = new Issue(getProjectById(id), summary); //argumentai turi buti pagal konstruktoriu, todel negalimpaduoti tik stringu
64          issues.add(issue);
65          return issue;
66      }
67
68      @Override
69      public Collection<Issue> getIssues(Project project) {
70          LinkedList<Issue> issuesByProject = new LinkedList<>();//sarasas metodo viduj, nes praso grazinti sarasa (o ne viena objekta)
71          for (Issue issue : issues) {
72              if (issue.getProject().equals(project)) {
73                  issuesByProject.add(issue);
74              }
75          }
76          return issuesByProject;
77      }
78
79      @Override
80      public Collection<Issue> getIssues(String id) {
81          LinkedList<Issue> issuesByProjectId = new LinkedList<>();//sarasas metodo viduj, nes praso grazinti sarasa (o ne viena objekta)
82          for (Issue issue : issues) {
83              if (issue.getProject().getId().equals(id)) {//isue turi projekta, o projektas turi id
84                  issuesByProjectId.add(issue);
85              }
86          }
87          return issuesByProjectId;
88      }
89  }
```