ASSIGNMENT-1

Question no : 07

7 Design and implement a console-based Bug Tracking application to create, assign, update, and

report software issues. Use OOP in Java.

Requirements:

1. Create at least 4 classes:

o User – id, name, role (QA/Dev/Manager), email.

o Issue – issueId, title, description, severity, status, assignee.

o Project – projectId, name, repoUrl, backlog (issues), team.

o TrackerService – createIssue/assign/update/status reports.

2. Each class must include:

o ≥4 instance/static variables.

o A constructor to initialize values.

o ≥5 methods (getters/setters, createIssue(), assignTo(), changeStatus(),

listBySeverity()).

3. Demonstrate OOPS Concepts:

o Inheritance → Manager extends User with override for approval.

o Method Overloading → createIssue() with/without attachments/tags.

o Method Overriding → custom display() in Issue subclasses (e.g., Bug, Task).

o Polymorphism → store issues as Issue and call overridden methods.

o Encapsulation → private fields, controlled updates.

4. Write a Main class (TrackerAppMain) to test:

o Create projects, users, issues.

o Assign issues, move NEW→IN_PROGRESS→RESOLVED→CLOSED.

o Print project dashboards and severity-wise reports.

CODE:

```
package tracker;


import java.util.*;

class User {
protected int id;
protected String name;
protected String role;
protected String email;

public User(int id, String name, String role, String email) {
this.id = id;
this.name = name;
this.role = role;
this.email = email;
}

public int getId() { return id; }
public String getName() { return name; }
public String getRole() { return role; }
public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }
```

```java
public void display() {
System.out.println(role + " " + name + " (" + email + ")");
}

public boolean approveIssue(Issue issue) {
System.out.println(role + " cannot approve issues.");
return false;
}
}

class Manager extends User {
public Manager(int id, String name, String email) {
super(id, name, "Manager", email);
}

public boolean approveIssue(Issue issue) {
if (issue.getStatus().equals("RESOLVED")) {
issue.setStatus("CLOSED");
System.out.println("Manager approved closure of Issue #" +
issue.getIssueId());
return true;
}
return false;
}
}
class Issue {
protected int issueId;
protected String title;
protected String description;
protected String severity;
protected String status;
protected User assignee;

public Issue(int issueId, String title, String description, String
severity) {
this.issueId = issueId;
this.title = title;
this.description = description;
this.severity = severity;
this.status = "NEW";
}

public int getIssueId() { return issueId; }
public String getTitle() { return title; }
public String getDescription() { return description; }
public String getSeverity() { return severity; }
public String getStatus() { return status; }
public User getAssignee() { return assignee; }
```

```java
public void setStatus(String status) { this.status = status; }
public void setAssignee(User u) { this.assignee = u; }

public void display() {
System.out.println("Issue #" + issueId + ": " + title + " [" +
severity + "] - " + status +
(assignee != null ? " (Assigned to " + assignee.getName() + ")" :
""));
}
}

class Bug extends Issue {
public Bug(int id, String title, String desc, String severity) {
super(id, title, desc, severity);
}

@Override
public void display() {
System.out.println("BUG #" + issueId + ": " + title + " - " +
severity + " - " + status);
}
}

class Task extends Issue {
public Task(int id, String title, String desc, String severity) {
super(id, title, desc, severity);
}

@Override
public void display() {
System.out.println("TASK #" + issueId + ": " + title + " - " +
severity + " - " + status);
}
}

class Project {
private int projectId;
private String name;
private String repoUrl;
private List<Issue> backlog = new ArrayList<>();
private List<User> team = new ArrayList<>();

public Project(int id, String name, String repoUrl) {
this.projectId = id;
this.name = name;
this.repoUrl = repoUrl;
}

public void addUser(User u) { team.add(u); }
```

```java
public void addIssue(Issue i) { backlog.add(i); }
public List<Issue> getIssues() { return backlog; }

public void displayDashboard() {
System.out.println("Project: " + name + " (" + repoUrl + ")");
for (Issue i : backlog) {
i.display();
}
}

public void listBySeverity(String severity) {
System.out.println("Issues with severity " + severity + ":");
for (Issue i : backlog)
if (i.getSeverity().equalsIgnoreCase(severity))
i.display();
}
}

class TrackerService {
private List<Project> projects = new ArrayList<>();

public void addProject(Project p) { projects.add(p); }

public Issue createIssue(Project p, int id, String title, String
desc, String severity) {
Issue i = new Bug(id, title, desc, severity); // default as Bug
p.addIssue(i);
return i;
}

public Issue createIssue(Project p, int id, String title, String
desc,
String severity, List<String> tags) {
Issue i = new Task(id, title + " " + tags, desc, severity);
p.addIssue(i);
return i;
}

public void assignTo(Issue i, User u) {
i.setAssignee(u);
i.setStatus("IN_PROGRESS");
System.out.println("Issue #" + i.getIssueId() + " assigned to " +
u.getName());
}

public void changeStatus(Issue i, String newStatus) {
i.setStatus(newStatus);
System.out.println("Issue #" + i.getIssueId() + " changed to " +
newStatus);
```

```java
}

public void reportBySeverity(Project p, String severity) {
p.listBySeverity(severity);
}
}

public class TrackerAppMain {
public static void main(String[] args) {
TrackerService service = new TrackerService();

User qa = new User(1, "Alice", "QA", "alice@mail.com");
User dev = new User(2, "Bob", "Dev", "bob@mail.com");
Manager mgr = new Manager(3, "Charlie", "charlie@mail.com");

Project proj = new Project(101, "BugTracker",
"https://github.com/demo");
proj.addUser(qa);
proj.addUser(dev);
proj.addUser(mgr);

service.addProject(proj);
Issue bug1 = service.createIssue(proj, 201, "Login fails", "Null
pointer on login", "HIGH");
Issue task1 = service.createIssue(proj, 202, "Update Docs", "Add API
usage docs",
"LOW", Arrays.asList("documentation"));

service.assignTo(bug1, dev);
service.assignTo(task1, qa);

service.changeStatus(bug1, "RESOLVED");
mgr.approveIssue(bug1); // Manager closes it

System.out.println("\n--- Project Dashboard ---");
proj.displayDashboard();

System.out.println("\n--- High Severity Issues ---");
service.reportBySeverity(proj, "HIGH");
}
}
```

OUTPUT :

Problems @ Javadoc Declaration Console

<terminated> TrackerAppMain [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.oper

```
Issue #201 assigned to Bob
Issue #202 assigned to Alice
Issue #201 changed to RESOLVED
Manager approved closure of Issue #201

--- Project Dashboard ---
Project: BugTracker (https://github.com/demo)
BUG #201: Login fails - HIGH - CLOSED
TASK #202: Update Docs [documentation] - LOW - IN_PROGRESS

--- High Severity Issues ---
Issues with severity HIGH:
BUG #201: Login fails - HIGH - CLOSED
```