public class Sprint {

private int capacity;

private int ticketsLimit;

private int currentEstimate;

Ticket[] tickets;

private int ticketCount;

public Sprint(int capacity, int ticketsLimit) {

this.capacity = capacity;

this.ticketsLimit = ticketsLimit;

tickets = new Ticket[ticketsLimit];

ticketCount = 0;

currentEstimate = 0;

}

public boolean addUserStory(UserStory userStory) {

if (userStory == null) return false;

if (userStory.getEstimate() + currentEstimate > capacity) return false;

if (ticketCount>=ticketsLimit) return false;

if (userStory.isCompleted()) return false;

// List<Ticket> accepted = Arrays.asList(tickets);

Ticket[] added = Arrays.copyOf(this.tickets, ticketCount);

for (UserStory dep: userStory.getDependencies()) {

if (!dep.isCompleted() && !isAdded(added, dep)) {

return false;

}

}

tickets[ticketCount] = userStory;

ticketCount ++;

currentEstimate+= userStory.getEstimate();

return true;

}

private boolean isAdded(Ticket[] added, Ticket ticket) {

for (Ticket t:added) {

if (t==ticket) {

return true;

}

}

return false;

}

public boolean addBug(Bug bugReport) {

if (bugReport == null) return false;

if (bugReport.getEstimate() + currentEstimate > capacity) return false;

if (ticketCount>=ticketsLimit) return false;

if (bugReport.isCompleted()) return false;

tickets[ticketCount] = bugReport;

ticketCount ++;

currentEstimate+= bugReport.getEstimate();

return true;

}

public Ticket[] getTickets() {

return Arrays.copyOf(this.tickets, ticketCount);

}

public int getTotalEstimate() {

int total = 0;

for (int i=0;i<ticketCount;++i) {

total += tickets[i].getEstimate();

}

return total;

}

}