abstract class TaskBase {  
 protected int id;  
 protected String name;  
 protected Date startDate;  
 protected Date endDate;  
 protected float progress;  
 protected TaskBase nextTask;  
  
 public int getId() {  
 return id;  
 }  
 public String getName() {  
 return name;  
 }  
 public Date getStartDate() {  
 return startDate;  
 }  
 public Date getEndDate() {  
 return endDate;  
 }  
 public float getProgress() {  
 return progress;  
 }  
 public void setProgress(float progress) {  
 this.progress = progress;  
 }  
 public void setNextTask(TaskBase nextTask) {  
 this.nextTask = nextTask;  
 }  
 public TaskBase getNextTask() {  
 return nextTask;  
 }  
  
 public TaskBase(int id, String name, Date startDate) {  
 this.id = id;  
 this.name = name;  
 this.startDate = startDate;  
 }  
  
 public boolean hasNextTask() {  
 return this.nextTask != null;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Task)) return false;  
  
 Task task = (Task) o;  
  
 return id == task.id;  
 }  
 @Override  
 public int hashCode() {  
 return id \* name.hashCode();  
 }  
}

public class Hito extends TaskBase {

public Hito(int id, String name, Date startDate) {

super(id, name, startDate);

}

}

abstract class TaskGroup extends TaskBase implements ICalculable {

public TaskGroup(int id, String name, Date startDate) {

super(id, name, startDate);

}

protected void closeTask(Date endDate) {

this.progress = 100;

this.endDate = endDate;

}

}

public class Task extends TaskGroup {

private List<Resource> resources;

public void addResources(Resource resource) {

this.resources.add(resource);

}

public List<Resource> getResources() {

return resources;

}

public Task(int id, String name, Date startDate, List<Resource> resources) {

super(id, name, startDate);

this.resources = resources == null ? new ArrayList<Resource>() : resources;

}

@Override

public void setNextTask(TaskBase nextTask) {

closeTask(nextTask.startDate);

super.setNextTask(nextTask);

}

@Override

public float getCost() {

float cost = 0;

for (Resource r : resources) {

cost += r.getCost();

}

return cost;

}

@Override

public Period getDuration() {

return Period.between(

LocalDate.ofEpochDay(getStartDate().getTime()),

LocalDate.ofEpochDay(getEndDate().getTime())

);

}

@Override

public String toString() {

return startDate +

"<" + name +

", progress=" + progress + "%" +

", resources=" + resources.size() +

", cost=" + this.getCost() +

">";

}

}

public class Summary extends TaskGroup {

public Summary(int id, String name, Task firstTask) {

super(id, name, firstTask.startDate);

this.nextTask = firstTask;

}

@Override

public float getCost() {

float cost = 0;

TaskBase t = nextTask;

while (t.hasNextTask()) {

cost += t.getCost();

}

return cost;

}

@Override

public Period getDuration() {

Period dur;

TaskBase t = nextTask;

Date startDateAux = t.startDate;

while (t.hasNextTask()) {

t = t.nextTask;

}

return Period.between(

LocalDate.ofEpochDay(startDateAux.getTime()),

LocalDate.ofEpochDay(t.endDate.getTime()));

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

TaskBase temp = this.nextTask;

while (temp.hasNextTask()) {

sb.append(temp.toString());

temp = nextTask.getNextTask();

}

return sb.toString();

}

}

public class Project extends TaskGroup {

private String description;

public Project(int id, String name, Summary firstTask) {

super(id, name, firstTask.startDate);

this.nextTask = firstTask;

}

@Override

public float getCost() {

return 0;

}

@Override

public Period getDuration() {

return null;

}

@Override

public String toString() {

return description;

}

}

abstract class Resource {

protected int id;

protected String name;

protected float cost;

public int getId() {

return id;

}

public String getName() {

return name;

}

protected abstract float getCost();

public Resource(int id, String name, float cost) {

this.id = id;

this.name = name;

this.cost = cost;

}

}

public class Programmer extends Resource {

@Override

public float getCost() {

return super.cost;

}

public Programmer(int id, String name, float cost) {

super(id, name, cost);

}

}