Repositories

UPskill

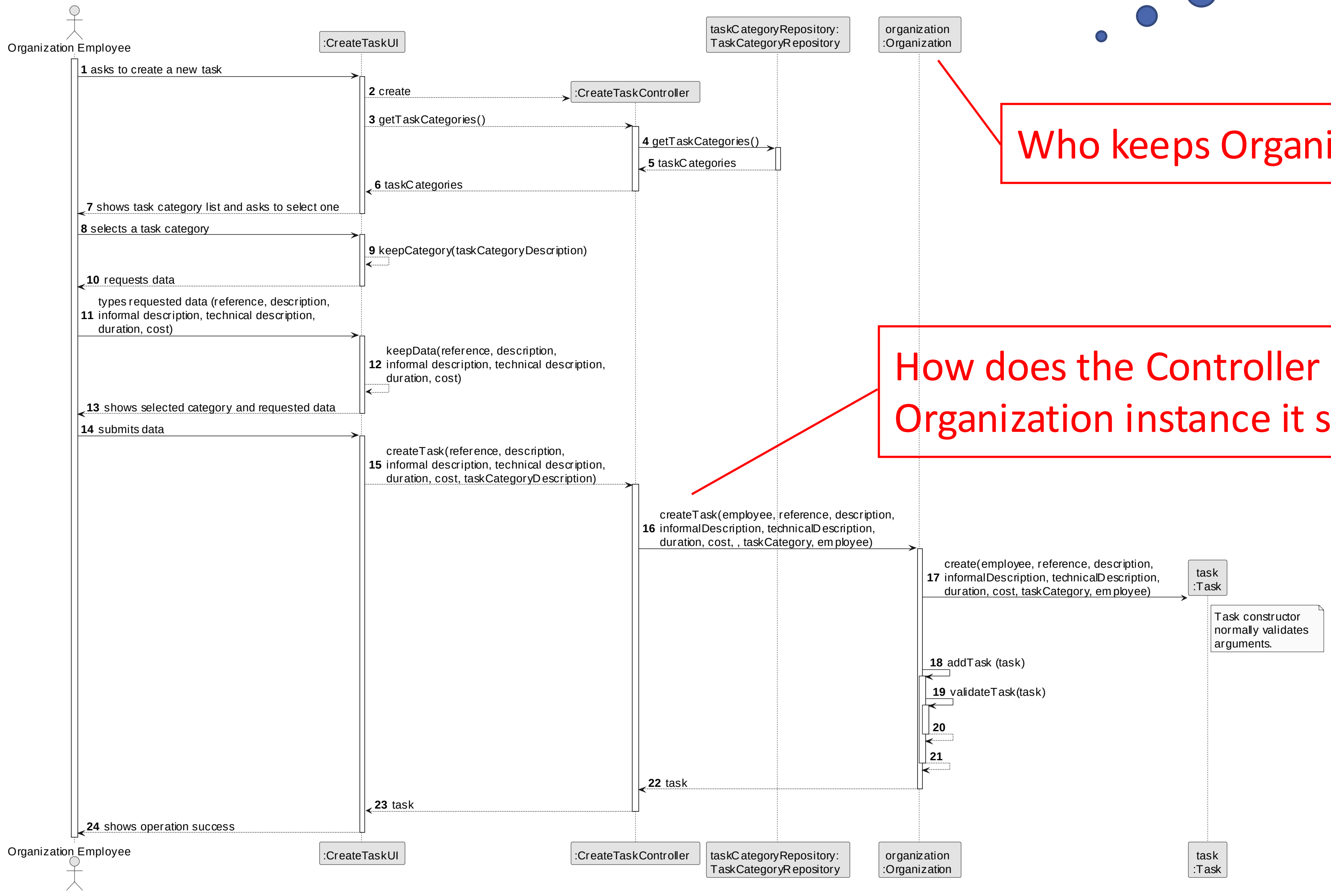
Digital Skills & Jobs

Topics

- Repositories
    - Organization Repository
- Singleton Pattern
    - Repositories class

Do you notice anything fishy?

Recall previous presentations



Organization Employee
:CreateTaskUI
:CreateTaskController
taskCategoryRepository: TaskCategoryRepository
organization :Organization
task :Task
1 asks to create a new task
2 create
3 getTaskCategories()
4 getTaskCategories()
5 taskCategories
6 taskCategories
7 shows task category list and asks to select one
8 selects a task category
9 keepCategory(taskCategoryDescription)
10 requests data
11 types requested data (reference, description, informal description, technical description, duration, cost)
12 keepData(reference, description, informal description, technical description, duration, cost)
13 shows selected category and requested data
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
17 create(employee, reference, description, informalDescription, technicalDescription, duration, cost, taskCategory, employee)
Task constructor normally validates arguments.
18 addTask (task)
19 validateTask(task)
20
21
22 task
23 task
24 shows operation success

Who keeps Organization instances?

How does the Controller know which Organization instance it should work with?

GRASP: Information Expert

Class OrganizationRepository

Information Expert

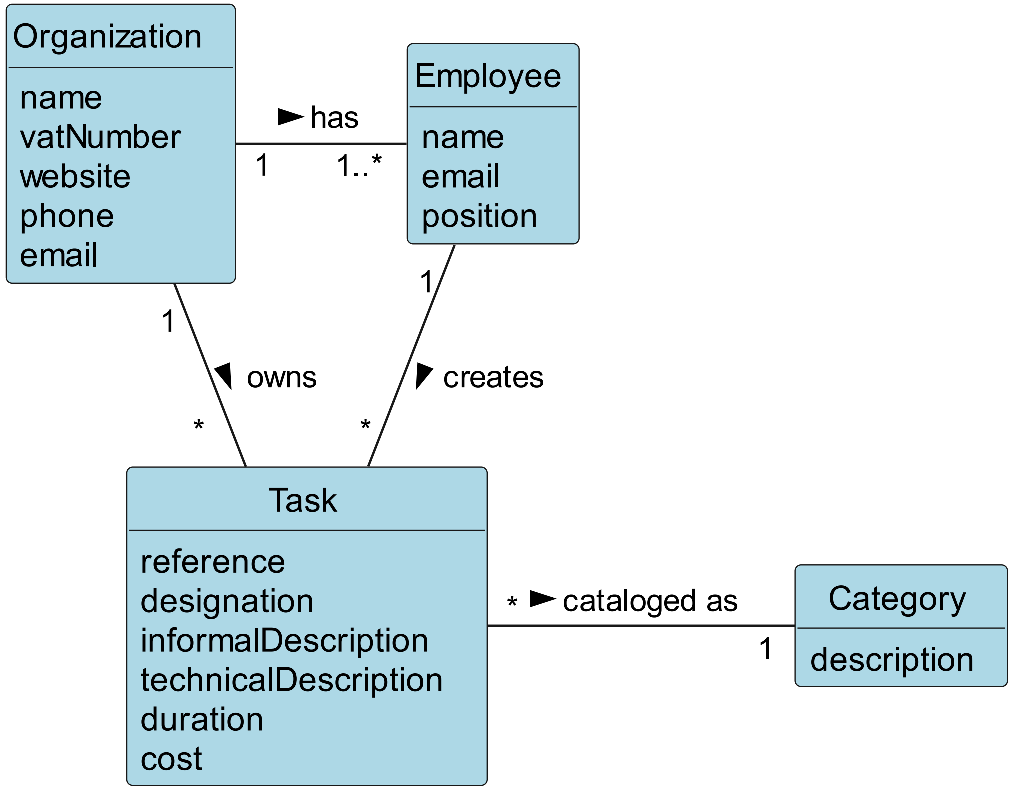
Recall previous presentations

- **Problem**
  - What is the **general principle** for assigning responsibilities to objects?

- **Solution:** Assign the responsibility to the **“information expert”**
  - I.e.: Assign it to the class that has the necessary information to fulfill it
  - Which class?
    - Get inspired by the **Domain Model**

Objects of the Domain Layer

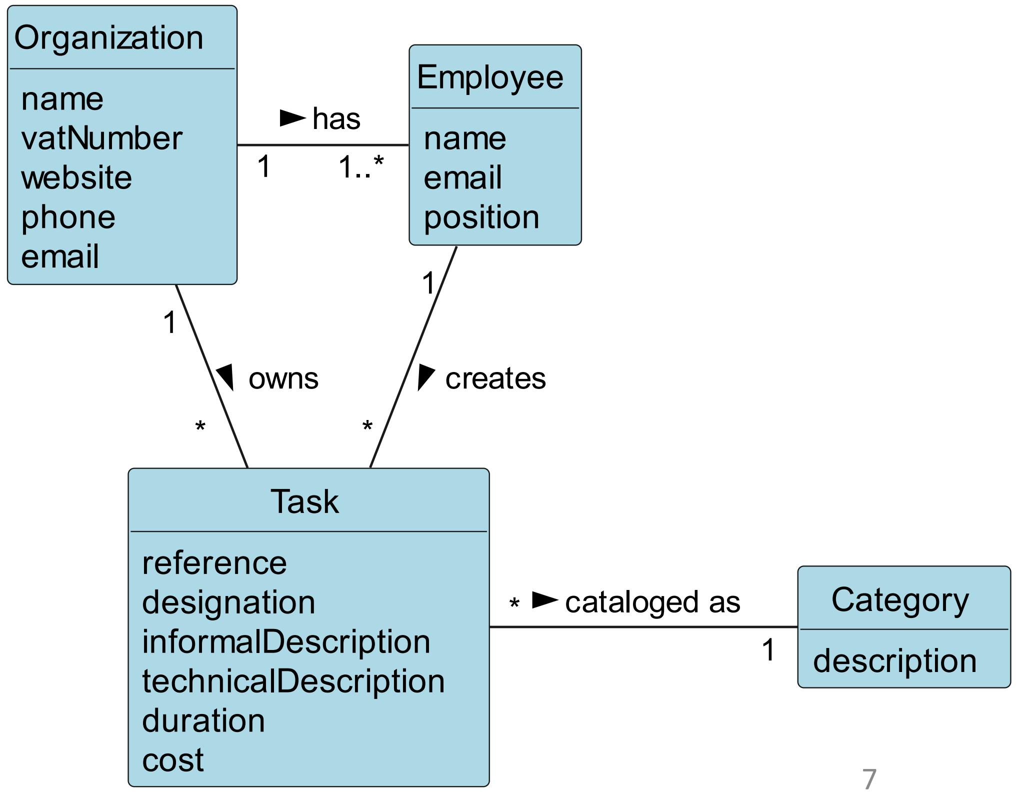
- **Who keeps Organization instances?**
  - Check the Domain Model...
- **Which class should be responsible for keeping instances of** a class that do not "belong" to any domain object/class, such as **Organization**?
  - Apparently, there is no conceptual class responsible for it.



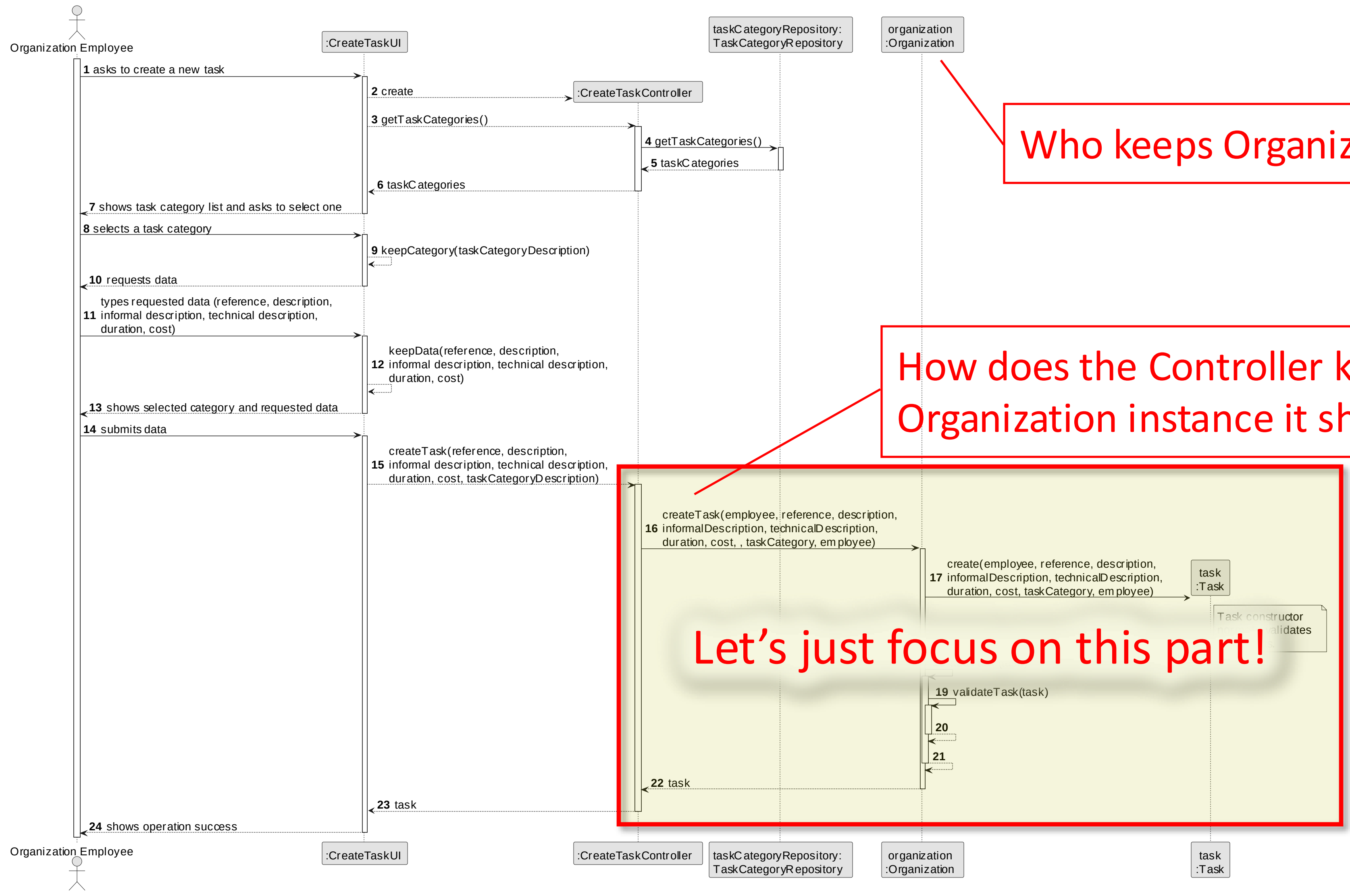
Repositories as Information Expert

- A **repository** is a special kind of **class for collection of objects that do not “belong” to any domain object/class**

- Looking at the domain model we realize that:
    - **Organizations are kept in the system** – they form a collection of objects that do not “belong” to any domain object/class
    - Repositories are also obtained by **Pure Fabrication**



Do you notice anything fishy?

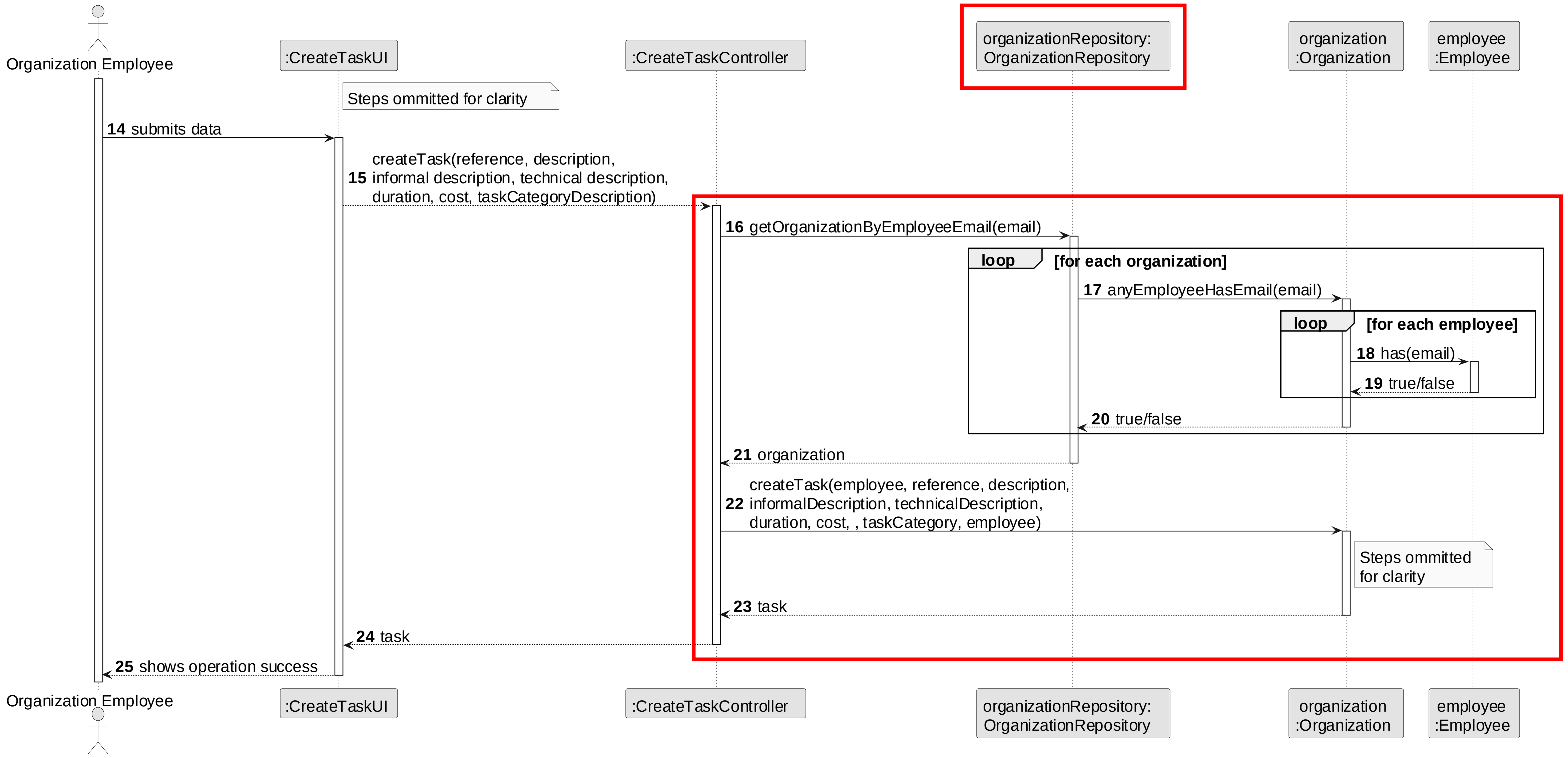

Organization Employee
:CreateTaskUI
:CreateTaskController
taskCategoryRepository:
TaskCategoryRepository
organization
:Organization
task
:Task
1 asks to create a new task
2 create
3 getTaskCategories()
4 getTaskCategories()
5 taskCategories
6 taskCategories
7 shows task category list and asks to select one
8 selects a task category
9 keepCategory(taskCategoryDescription)
10 requests data
11 types requested data (reference, description, informal description, technical description, duration, cost)
12 keepData(reference, description, informal description, technical description, duration, cost)
13 shows selected category and requested data
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
17 create(employee, reference, description, informalDescription, technicalDescription, duration, cost, taskCategory, employee)
Task constructor
validates
19 validateTask(task)
20
21
22 task
23 task
24 shows operation success

Who keeps Organization instances?

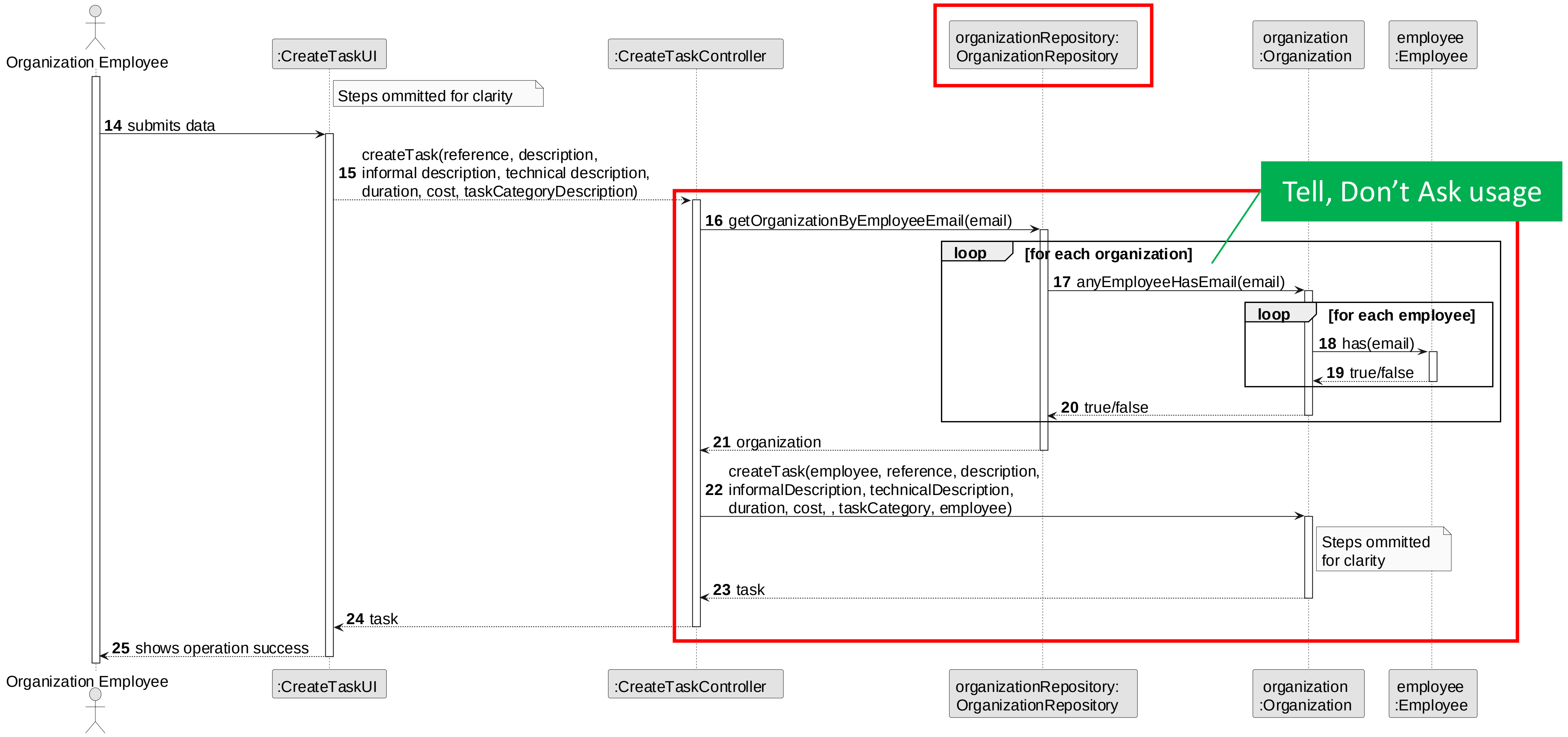
How does the Controller know which Organization instance it should work with?

Let's just focus on this part!

OrganizationRepository responsibilities (1/4)

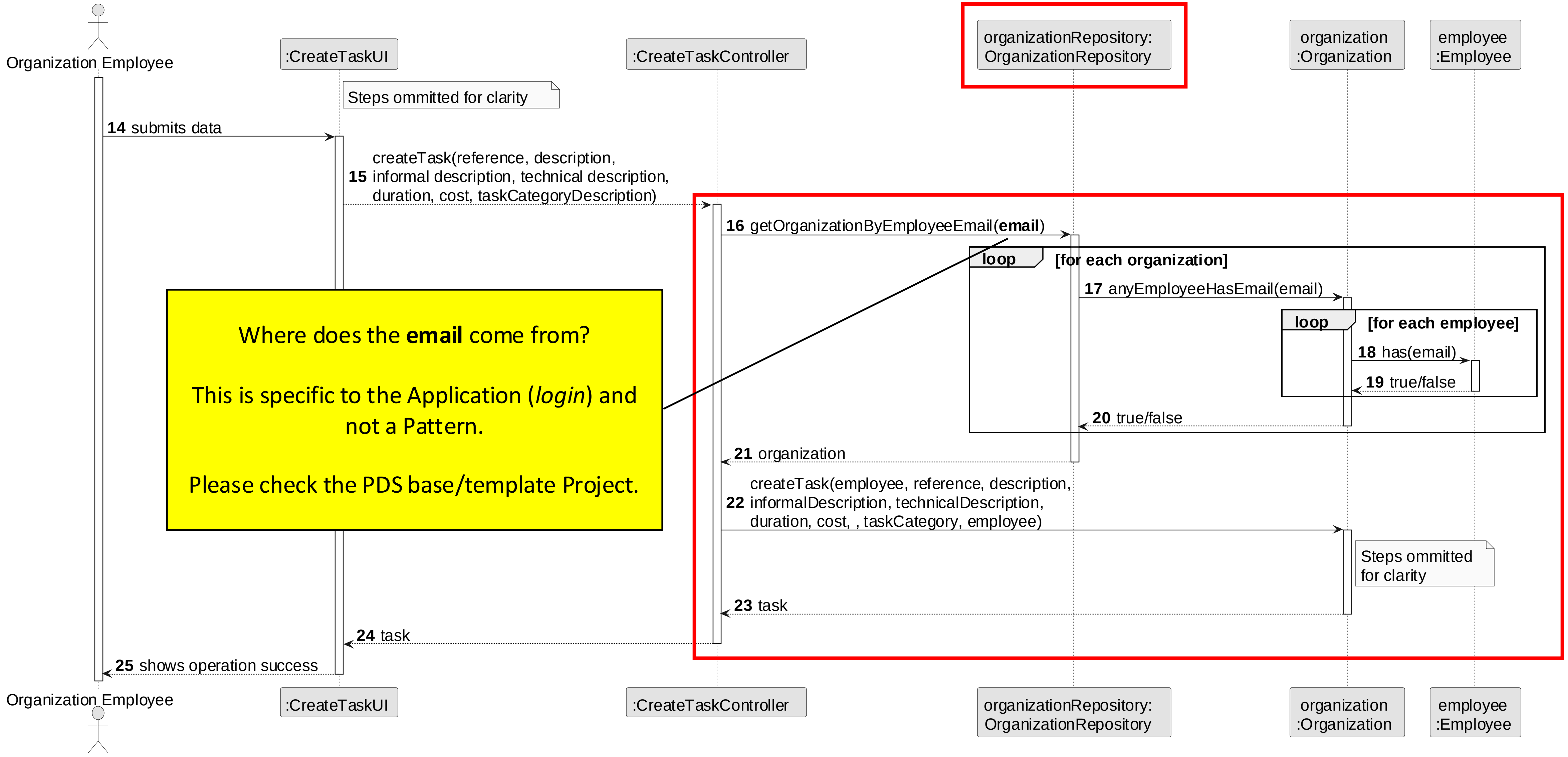

Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getOrganizationByEmployeeEmail(email)
loop [for each organization]
17 anyEmployeeHasEmail(email)
loop [for each employee]
18 has(email)
19 true/false
20 true/false
21 organization
22 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
23 task
24 task
25 shows operation success
Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee

OrganizationRepository responsibilities (2/4)



Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getOrganizationByEmployeeEmail(email)
loop [for each organization]
17 anyEmployeeHasEmail(email)
loop [for each employee]
18 has(email)
19 true/false
20 true/false
21 organization
22 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
23 task
24 task
25 shows operation success
Tell, Don't Ask usage

OrganizationRepository responsibilities (3/4)



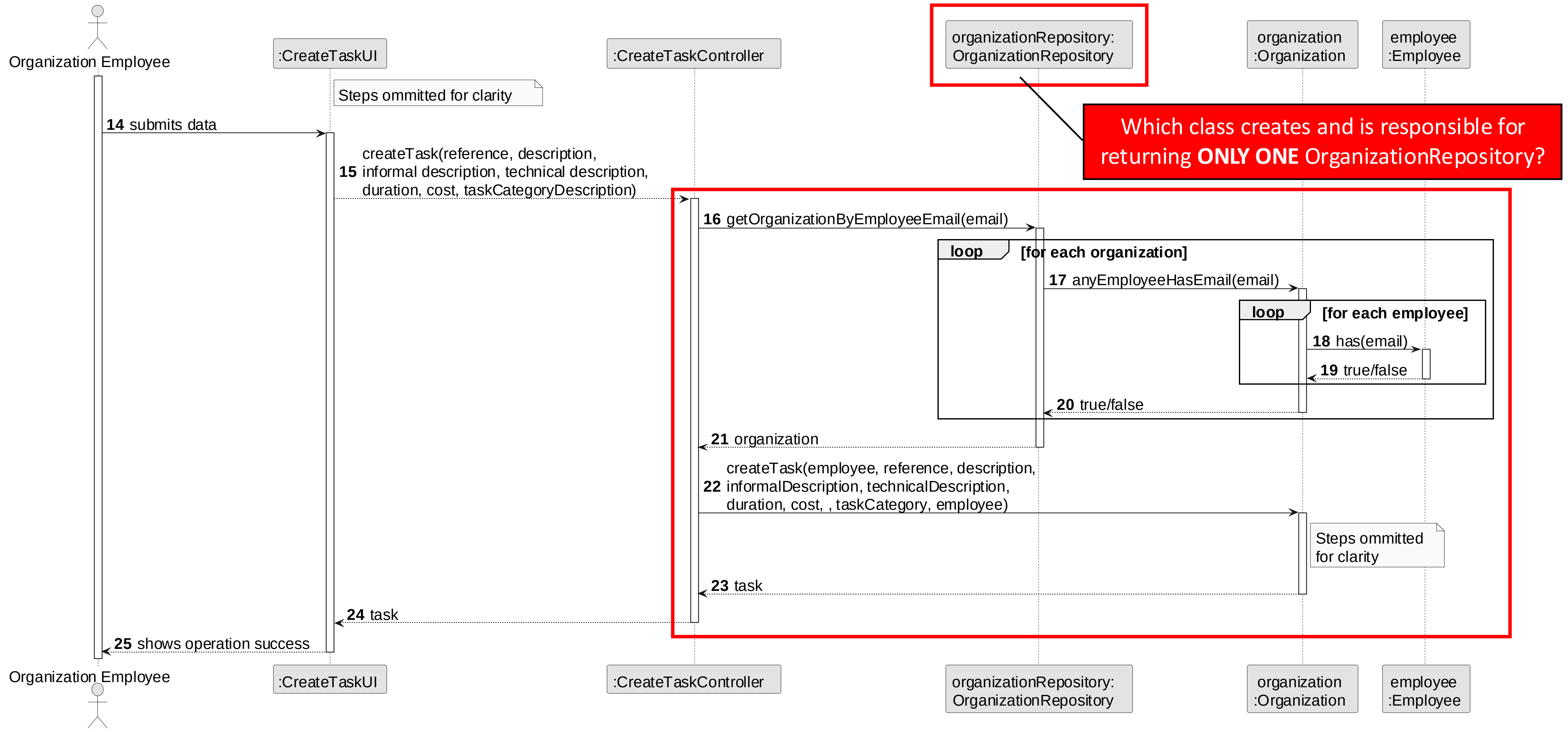
Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getOrganizationByEmployeeEmail(email)
loop [for each organization]
17 anyEmployeeHasEmail(email)
loop [for each employee]
18 has(email)
19 true/false
20 true/false
21 organization
22 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
23 task
24 task
25 shows operation success

Where does the **email** come from?

This is specific to the Application (*login*) and not a Pattern.

Please check the PDS base/template Project.

OrganizationRepository responsibilities (4/4)


Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getOrganizationByEmployeeEmail(email)
loop [for each organization]
17 anyEmployeeHasEmail(email)
loop [for each employee]
18 has(email)
19 true/false
20 true/false
21 organization
22 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
23 task
24 task
25 shows operation success

Which class creates and is responsible for returning **ONLY ONE** OrganizationRepository?

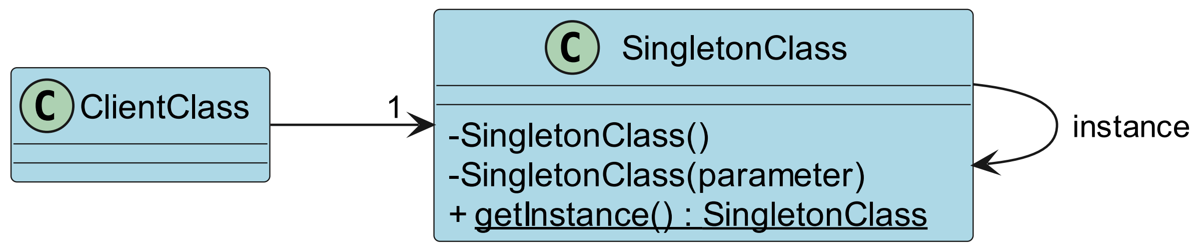
Singleton Pattern

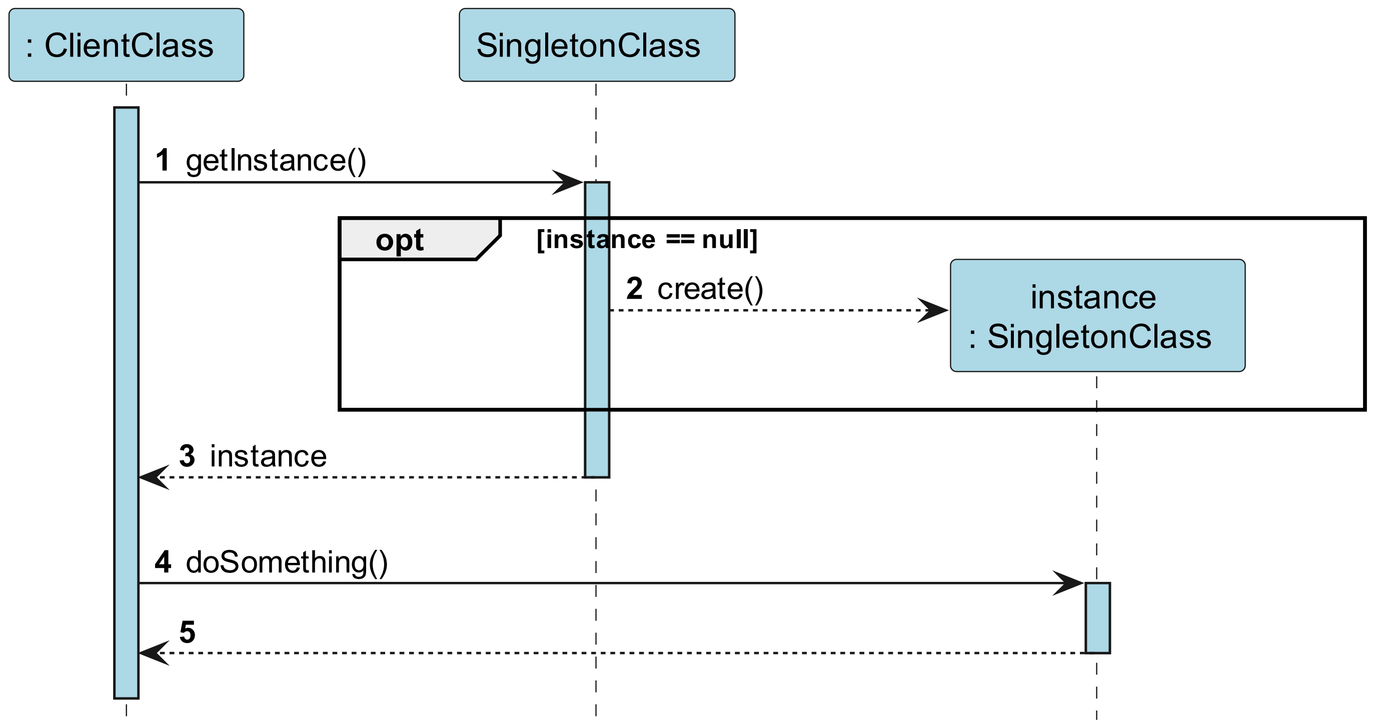
Applied to Repositories class

Singleton Pattern

- **Problem**
  - How to ensure that there is **only one instance** of a given class; and
  - How to provide a global access point to such instance?
- **Solution**
  - In a very simple way, the *singleton* class needs to:
    - Have its **constructor(s) private** (or protected), to prevent other classes from creating instances
    - Have a **private static attribute to itself**
    - Have a **public static method** (usually called ***getInstance***) that is used by other classes to get the unique instance of the class
  - Instance creation
    - ***Lazy*** *initialization*: restricts the creation of the class instance until it is requested for the first time
    - ***Eager*** *initialization*: the class instance is created during the start-up time

Singleton Pattern – *Lazy initialization*





Singleton Pattern – Overview

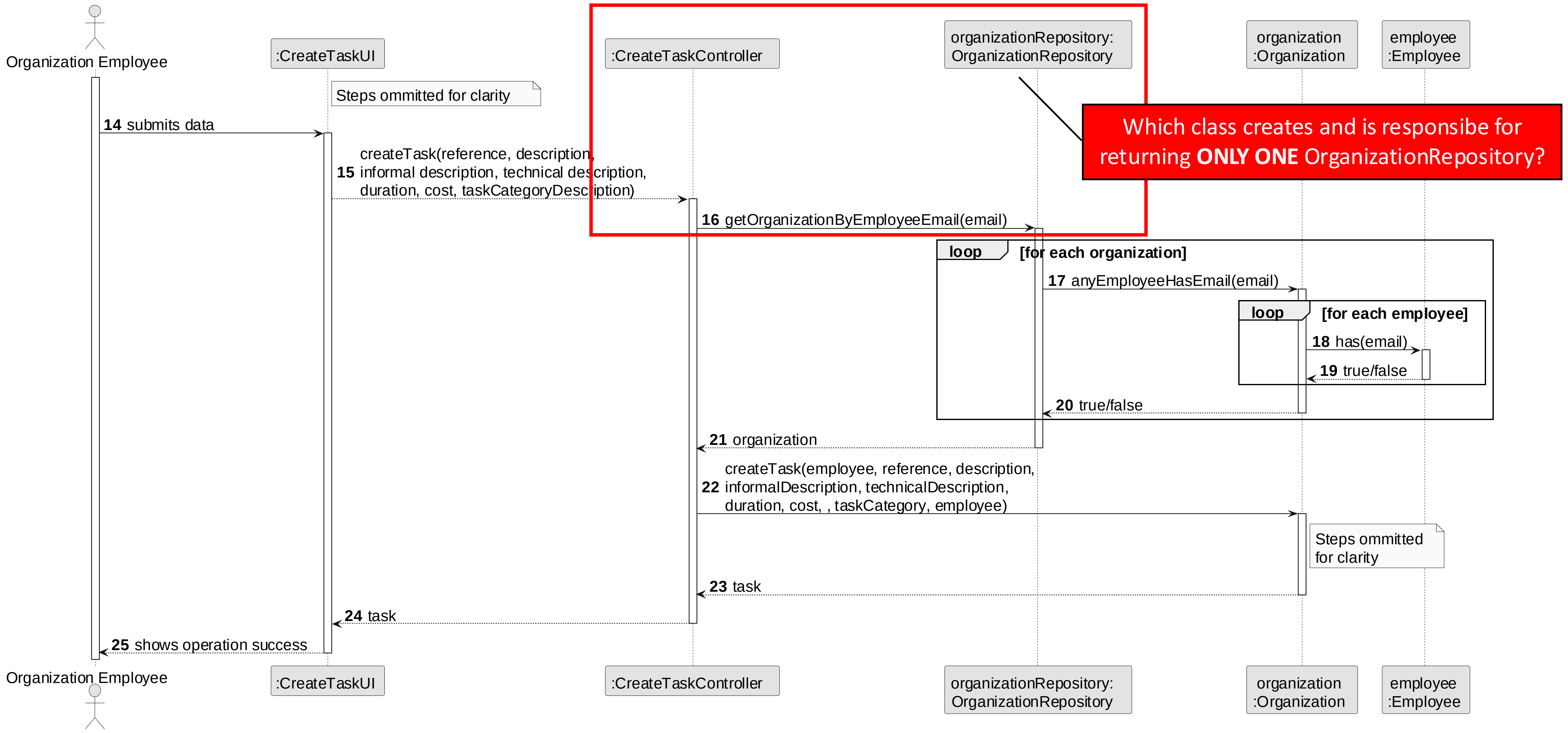
- The Singleton pattern is a well-known software design pattern
  - From the “Gang of Four (GOF) Design Patterns”
- **Singleton syndrome:** it is too often seen as the most appropriate pattern for your current use case → But it turns out it is not!
- Sometimes, it is considered an **Anti-Pattern**
  - Its usage should be **minimized**, as it promotes the existence of an application global state and testing activities become harder.
  - There are better techniques than using it, but those are more advanced and beyond the scope of this course. Next year you will learn and use them in other course units.

UPskill

Repositories class

- In the PDS base/template Project, the **Repositories class** is implemented using the Singleton Pattern
- This class **provides access to all data repositories** that may be required by the software solution, thus **minimizing the usage** of the Singleton pattern
- It ALWAYS provides the **same and unique** repositories instance

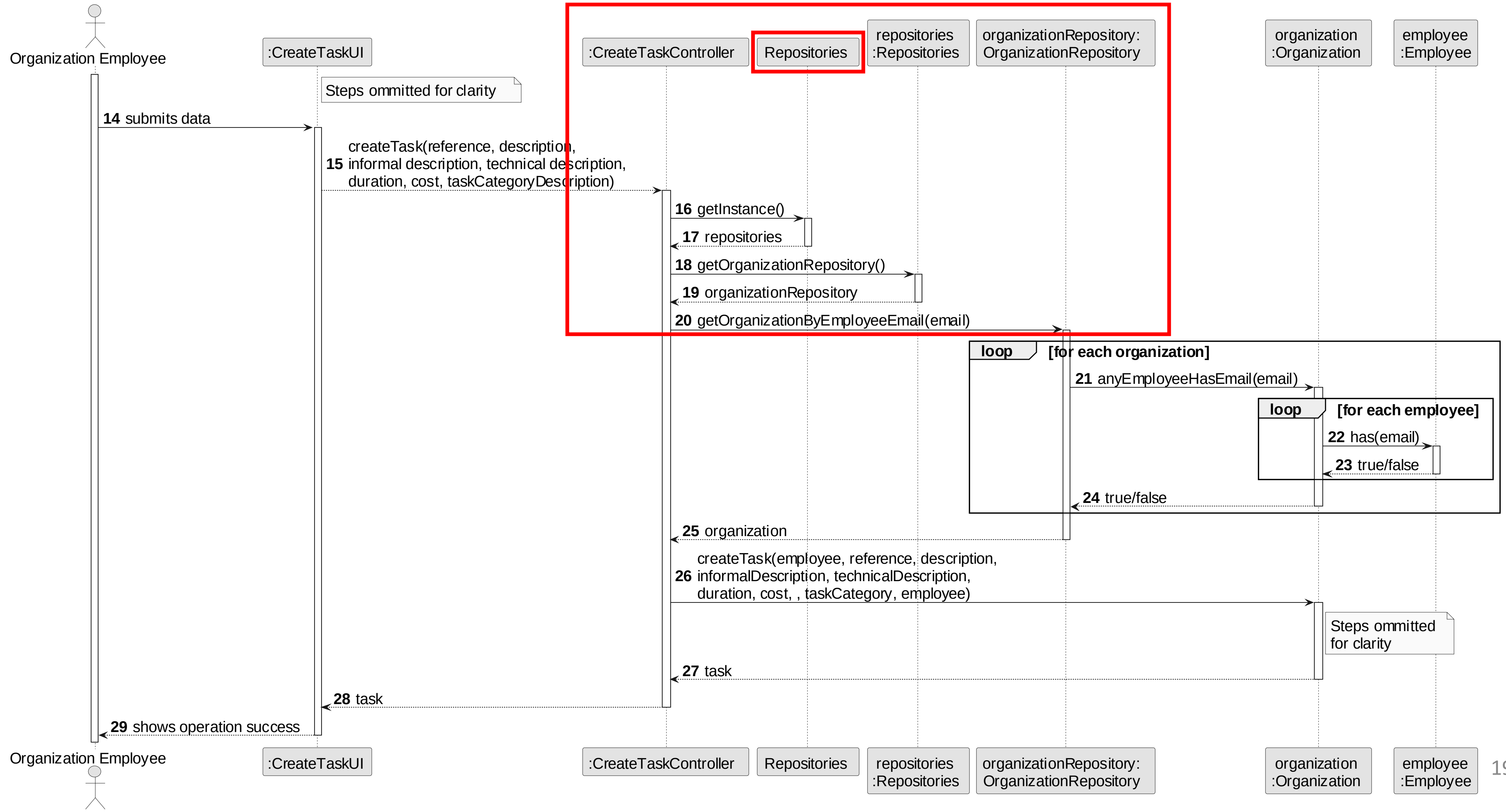
Repositories usage (1/2)



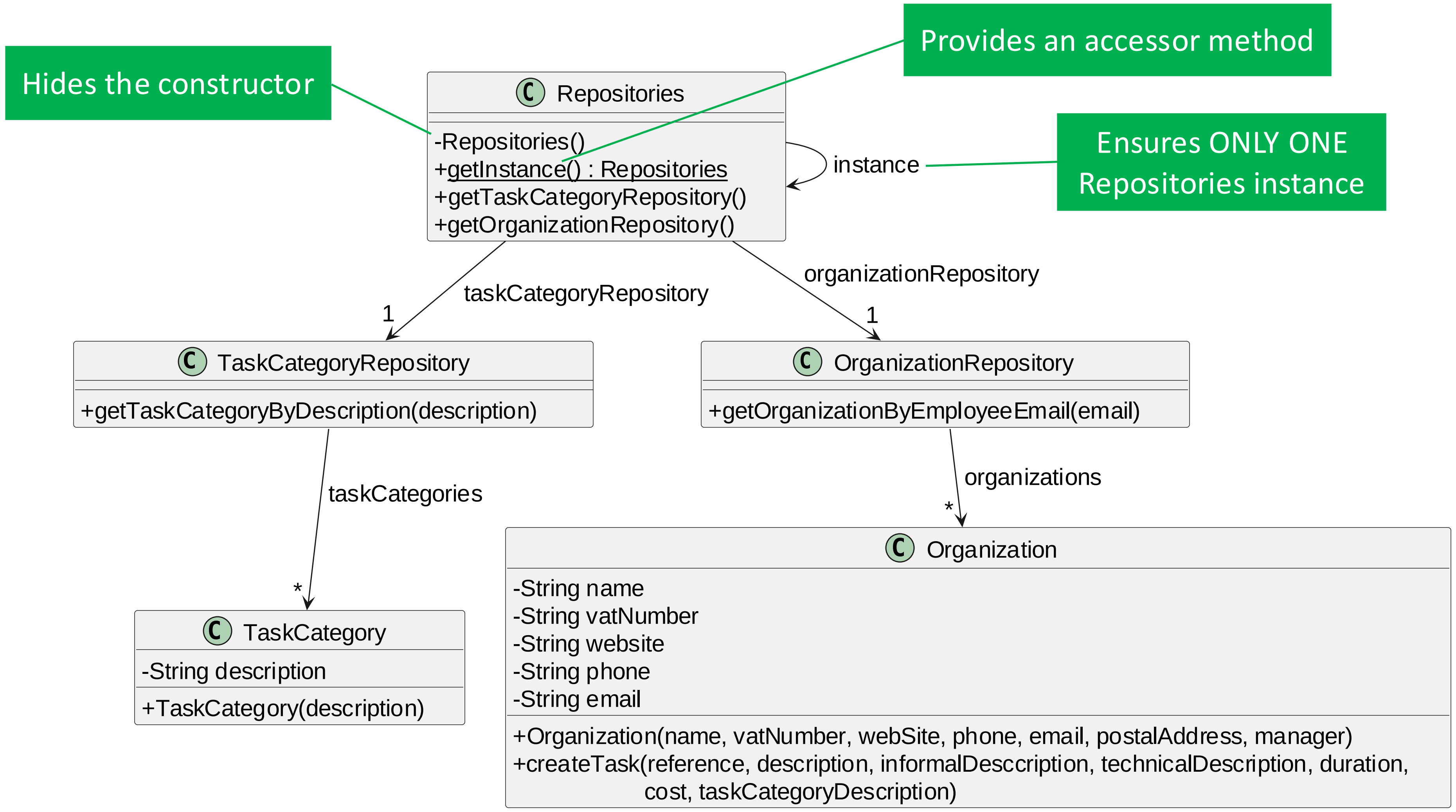
Organization Employee
:CreateTaskUI
:CreateTaskController
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getOrganizationByEmployeeEmail(email)
loop [for each organization]
17 anyEmployeeHasEmail(email)
loop [for each employee]
18 has(email)
19 true/false
20 true/false
21 organization
22 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
23 task
24 task
25 shows operation success

Which class creates and is responsibe for returning **ONLY ONE** OrganizationRepository?

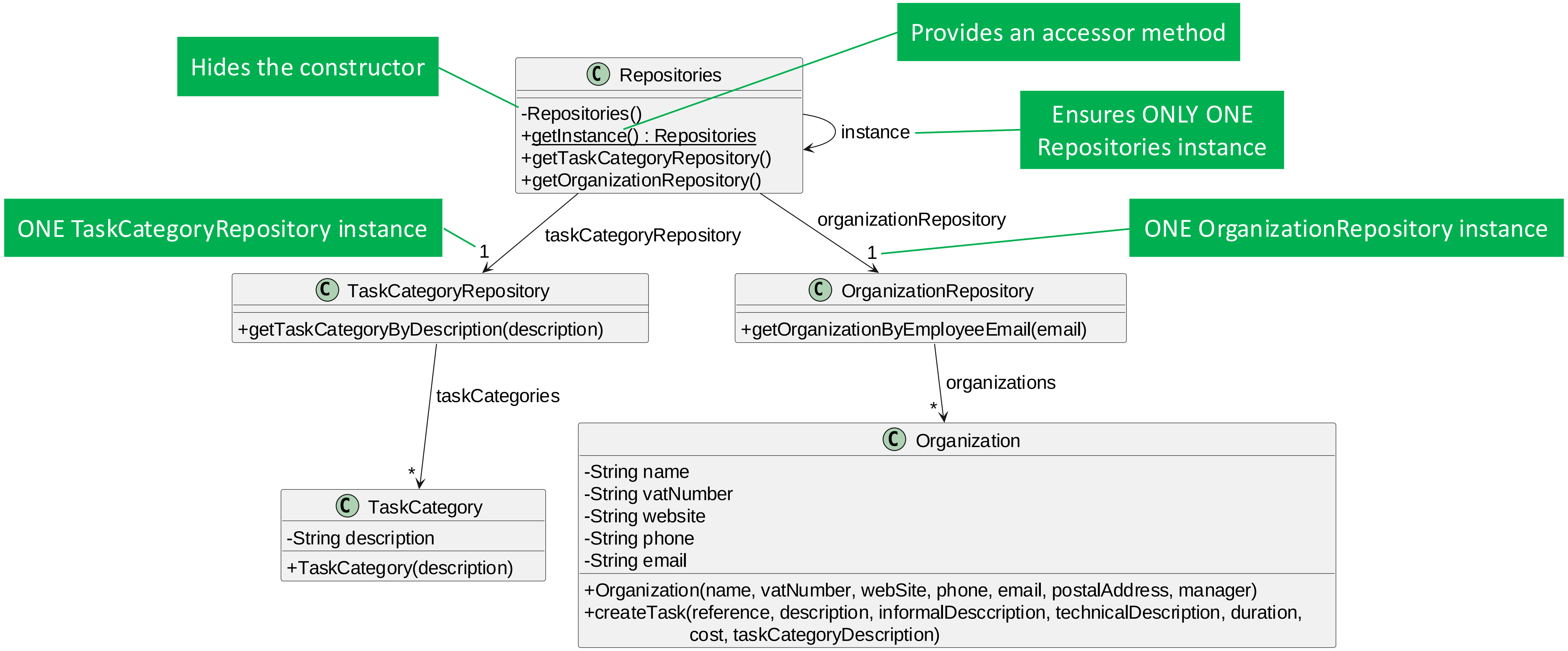
Repositories usage (2/2)


Organization Employee
:CreateTaskUI
:CreateTaskController
Repositories
repositories :Repositories
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee
Steps ommitted for clarity
14 submits data
15 createTask(reference, description, informal description, technical description, duration, cost, taskCategoryDescription)
16 getInstance()
17 repositories
18 getOrganizationRepository()
19 organizationRepository
20 getOrganizationByEmployeeEmail(email)
loop [for each organization]
21 anyEmployeeHasEmail(email)
loop [for each employee]
22 has(email)
23 true/false
24 true/false
25 organization
26 createTask(employee, reference, description, informalDescription, technicalDescription, duration, cost, , taskCategory, employee)
Steps ommitted for clarity
27 task
28 task
29 shows operation success
Organization Employee
:CreateTaskUI
:CreateTaskController
Repositories
repositories :Repositories
organizationRepository: OrganizationRepository
organization :Organization
employee :Employee

Repositories Class Diagram (1/2)


Hides the constructor
Provides an accessor method
Ensures ONLY ONE Repositories instance
Repositories
-Repositories()
+getInstance() : Repositories
+getTaskCategoryRepository()
+getOrganizationRepository()
instance
taskCategoryRepository
1
organizationRepository
1
TaskCategoryRepository
+getTaskCategoryByDescription(description)
OrganizationRepository
+getOrganizationByEmployeeEmail(email)
taskCategories
*
organizations
*
TaskCategory
-String description
+TaskCategory(description)
Organization
-String name
-String vatNumber
-String website
-String phone
-String email
+Organization(name, vatNumber, webSite, phone, email, postalAddress, manager)
+createTask(reference, description, informalDesccription, technicalDescription, duration, cost, taskCategoryDescription)

Repositories Class Diagram (2/2)


Hides the constructor
Provides an accessor method
Ensures ONLY ONE Repositories instance
ONE TaskCategoryRepository instance
ONE OrganizationRepository instance
Repositories
-Repositories()
+getInstance() : Repositories
+getTaskCategoryRepository()
+getOrganizationRepository()
instance
taskCategoryRepository
organizationRepository
1
1
TaskCategoryRepository
+getTaskCategoryByDescription(description)
OrganizationRepository
+getOrganizationByEmployeeEmail(email)
taskCategories
organizations
*
*
TaskCategory
-String description
+TaskCategory(description)
Organization
-String name
-String vatNumber
-String website
-String phone
-String email
+Organization(name, vatNumber, webSite, phone, email, postalAddress, manager)
+createTask(reference, description, informalDesccription, technicalDescription, duration, cost, taskCategoryDescription)

References & Bibliography

- Larman, Craig; Applying UML and Patterns; Prentice Hall (3rd ed.); ISBN 978-0131489066
- Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.
- Rational Unified Process: Best Practices for Software Development Teams; Rational Software White Paper; TP026B, Ver 11/01.
- https://www.kamilgrzybek.com/blog/posts/grasp-explained
- https://en.wikipedia.org/wiki/Singleton_pattern