Tracker Project

Name: Bryan Brkic
Student Number: 213958806
Prism Login: brkicb

Signature: 

Table of Contents

1. Software Product Requirements

A tracker system is used for monitoring waste products in a nuclear plant and makes sure that they are handled in a safe manner. Operators of the tracker system should be able to safely manage radioactive waste in the nuclear plant.

Containers holding these waste materials pass through the power plant. These containers go through different stages of processing during tracking. Within the tracking plant are several phases which handle the radioactive materials and not all nuclear plants will have the same phases.

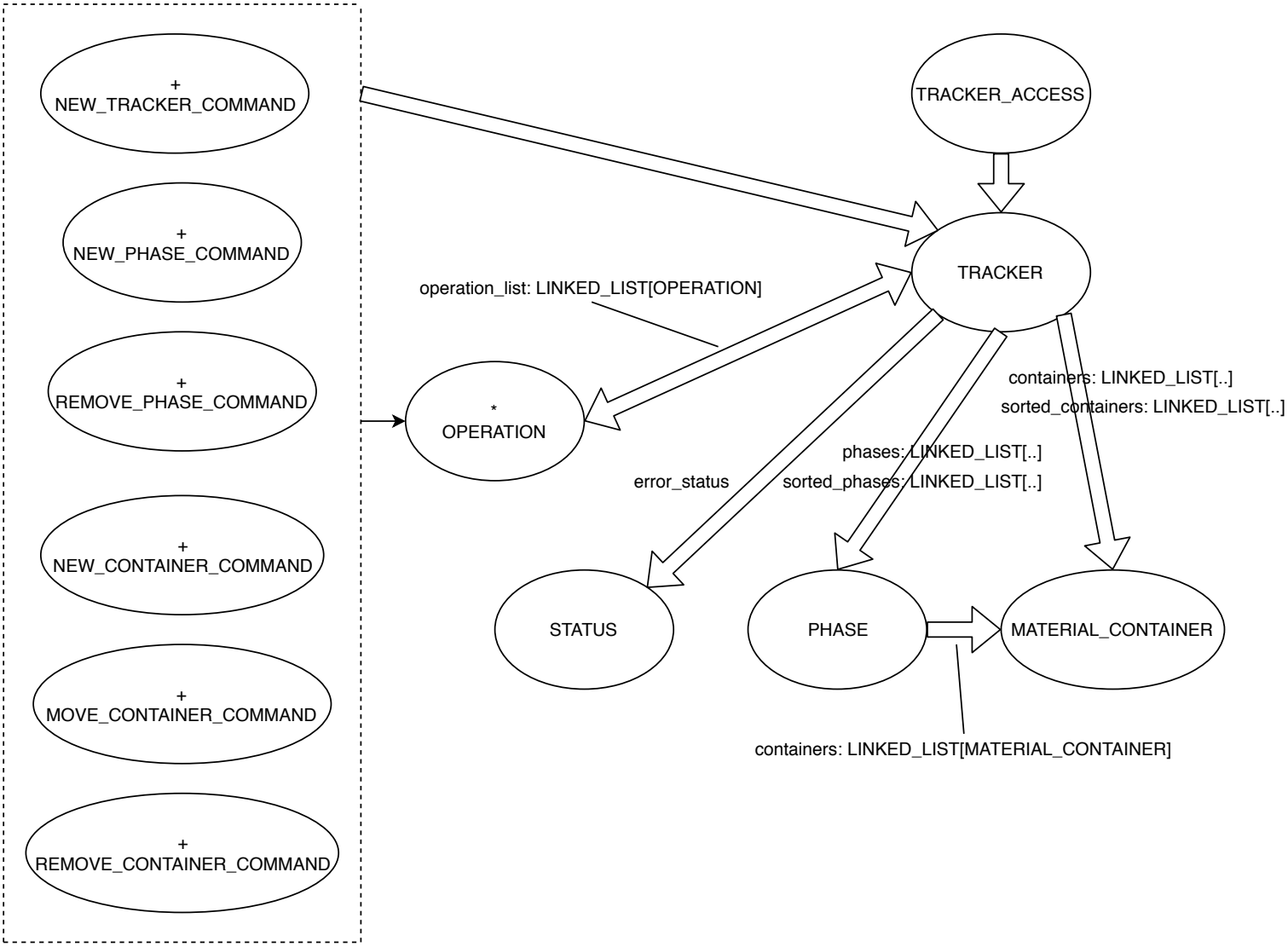
Every plant will have a maximum radiation amount for a phase in the process and for a container in the system. If a container is being placed in the system and will exceed the maximum radiation amount, then the tracker system should signal an error.

Operations for the tracker system include NEW_TRACKER_COMMAND, which creates a new tracker with a maximum phase radiation and maximum container radiation being set. NEW_PHASE_COMMAND adds a new phase into the tracker system, and to create a new a phase it needs; a phase ID, a name for the phase, the maximum number of containers that can be in the phase, and the materials that can be in the phase. REMOVE_PHASE_COMMAND removes a phase from the system and to remove a phase you need to pass in the phase ID of the phase you want to remove. NEW_CONTAINER_COMMAND adds a new container into a phase. To add a new container you require; a container ID for the container, the material in the container, the amount of radiation in the container, and the phase ID of the phase the container will be placed in. REMOVE_CONTAINER_COMMAND removes a container from the system and requires the container ID to be passed for the container to be removed. MOVE_CONTAINER_COMMAND is used for moving a container from one phase to another. In order to move a container you require; the container ID of the container you want to move, the phase ID of the phase that the container is within, and the phase ID you want to move the container to. The tracker system should be able to perform all these commands and be able to handle errors that may occur for each individual command.

Being able to create a new tracker system with phases correctly and be able to remove phases from the system satisfies the first goal which is the setup. Being able to add new containers into a phase correctly and be able to both move a container from one phase to another and be able to remove a container satisfies the second goal which is the operation. Finally, being able to handle errors correctly, display proper error messages, and being able to monitor correctly various conditions within the tracker system satisfies the third goal which is safety.

2. BON class diagram overview (*architecture* of the design)

a)



b)

TRACKER is the class that handles the overall functionality of the tracker system. By default when the program runs, you have a tracker that has a maximum phase radiation set a 0.0 and maximum container radiation set at 0.0. As a user you have eight different commands you can run. The first is new_tracker is handled by the NEW_TRACKER_COMMAND class which allows you to create a new tracker system where you set a maximum phase radiation and maximum container radiation. The second is new_phase which is handled by the NEW_PHASE_COMMAND class which allows you to add a new phase into the tracker system. The third is remove_phase which is handled by the REMOVE_PHASE_COMMAND class which removes a phase from the tracker system. The fourth is new_container which is handled by the NEW_CONTAINER_COMMAND class which adds a container into a phase in the tracker system. The fifth is move_container which is handled by the MOVE_CONTAINER_COMMAND class which moves a container from one phase to another. The sixth is remove_container which is handled by the REMOVE_CONTAINER_COMMAND class which removes a container from a phase in the tracker system. Finally we have the seventh and eighth commands which are undo and redo, and each of the previous commands have to be able to handle these two commands correctly.

In my tracker system I decided to have a class PHASE for representing a phase in the system, and MATERIAL_CONTAINER for representing a container. I decided to do things this way since it was the easiest way to manage what containers are in a particular phase. Also this way you can check easily for various errors such as a container being present in two different phases. Each phase has a linked list for holding the material containers, and this linked list is what's used to check for containers in a phase. Then the tracker system itself has a linked list called phases which holds the phases in the tracker system. Then the tracker system also has a linked list called containers for holding a list of containers in the system which is useful for keeping track of every container in the system and easily sorting it to display containers for the user. Then we have sorted_phases which holds a list of phases in sorted order, and sorted_containers which holds a list of containers in sorted order which is used for displaying phases and containers to the user in sorted order.

We also have the STATUS class which holds different status/error messages that get displayed. In this class we have an ok status which is used when things are in order, and then we have 20 different error messages used for the various errors that can occur within the tracker system. The STATUS class also has an is_empty attribute used for checking if there is an error or if everything is in order, and is useful for correctly displaying to the user.

3. Table of modules — responsibilities and secrets

1	TRACKER	**Responsibility**: Represents the current state of a tracker system.	**Alternative**: none
	Concrete	**Secret**: Includes information about phases in the system and containers in the system, the current state number, the current status, and various checks used for displaying correctly to the user.	

1.1	PHASE	**Responsibility**: Represents a phase in the tracker system.	**Alternative**: none
	Concrete	**Secret**: Holds information about the phase ID, phase name, the maximum amount of containers that can be held, a list of materials that can be present in the phase, and a list of containers within the phase.	

1.2	MATERIAL_CONTAINER	**Responsibility**: Represents a container within a tracker system and more precisely a container within a phase.	**Alternative**: none
	Concrete	**Secret**: Holds information about the container ID, material ID, the material the container holds, and the amount of radiation.	

1.3	STATUS	**Responsibility**: Where you have access to different possible display messages which include an ok_status and 20 different error messages.	**Alternative**: none
	Concrete	**Secret**: Holds information about whether there is an error or if everything is fine in the system using the is_empty attribute, and this helps for displaying correctly to the user.	

1.4	OPERATION	**Responsibility**: Represents an operation in a tracker system.	**Alternative**: none
	Abstract	**Secret**: none	

1.4.1	NEW_TRACKER_COMMAND	**Responsibility**: Operation used for creating a new tracker system.	**Alternative**: none
	Concrete	**Secret**: none	

1.4.2	NEW_PHASE_COMMAND	**Responsibility**: Operation used for a creating a new phase in the tracker system.	**Alternative**: none
	Concrete	**Secret**: none	

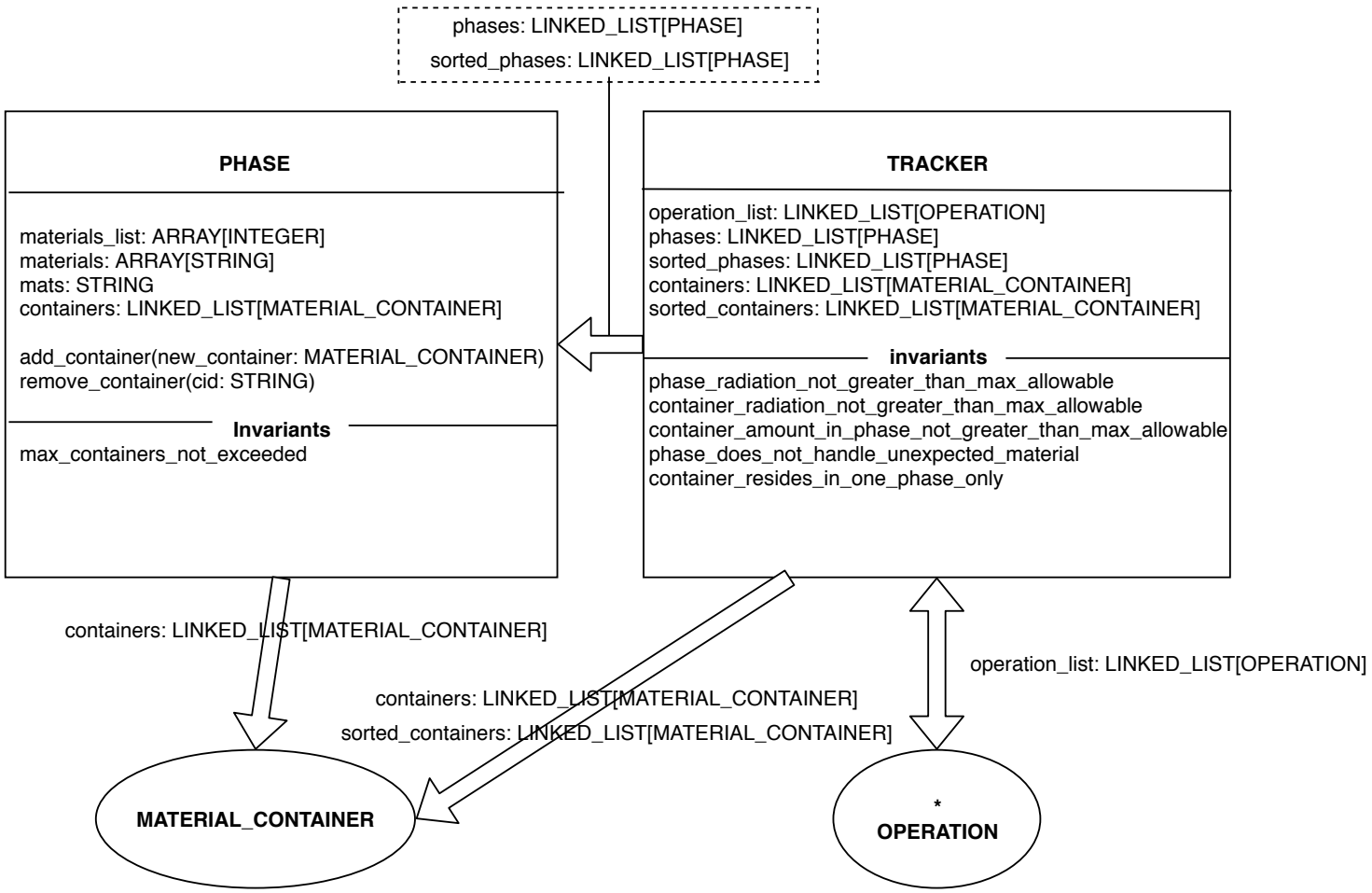
1.4.3	NEW_CONTAINER_COMMAND	**Responsibility**: Operation used for creating a new container to a phase in the system.	**Alternative**: none
	Concrete	**Secret**: none	

1.4.4	REMOVE_PHASE_COMMAND	**Responsibility**: Operation used for removing a phase from the tracker system.	**Alternative**: none
	Concrete	**Secret**: none	

1.4.5	MOVE_CONTAINER_COMMAND	**Responsibility**: Operation used for moving a container from one phase to another.	**Alternative**: none
	Concrete	**Secret**: none	

1.4.6	REMOVE_CONTAINER_COMMAND	**Responsibility**: Operation used for removing a container from a phase in the track system.	**Alternative**: none
	Concrete	**Secret**: none	

4. Expanded description of design decisions



Within the tracker system I have various invariants that are used to satisfy the third design goal which deals with the safety of the system. The diagram shows more in depth how I handle the phases and containers within the tracker system. In the tracker system itself, I have a linked list called phases for storing the phases in the system. The linked list sorted_phases is used for printing to the user the phases in sorted order. Once things get to the containers, things get trickier since each phase holds a list of containers it contains. In order to print the containers easily to the user in sorted order, I used a linked list containers to keep track of all containers in the system, and the linked_list sorted_containers is used to print the containers in sorted order to the user. Thus the linked list containers and sorted_containers are only used for the printing of containers to the user, but for having correct logic in our program and avoid errors, we always use the linked list containers held in the PHASE class, that way we can check things such as if a container is in two different phases. By having a PHASE class that

holds a linked list of containers it contains, it makes these logical checks very simple, and with the linked list containers and sorted containers in the TRACKER class we have an easy way to print to the user and therefore have an easy way to keep track of containers in the tracker system. I just had to ensure when a container is added or removed that the linked list is correctly updated in the PHASE class and in the TRACKER class. Then for moving a container we don't have to worry about the linked list containers in the TRACKER class since the container will remain in the system and this linked list only worries about keeping track of all containers in the system. To make adding and removing of containers in a PHASE easier, I added a procedure to add a container in the phase and one to remove a container in the phase. Then I also have an invariant in the phase class for ensuring that it's maximum container capacity is not exceeded. There is also an invariant in the TRACKER class that does this check for all phases, so it's not entirely necessary, but it may still help with correctness in the tracker system.

Then we have an attribute operation_list in the TRACKER class which is implemented using a linked list and keeps track of all the operations that are performed. This is useful for the functionality of the undo and redo commands. The redo command functions the same for all the operations. We just store the state number of the operation in a local variable, then we run the execute command, then we set the state number to the value in our local variable to preserve its value. As for the undo command, it is implemented slightly differently for each command. For the new_tracker command, if it was run correctly, it will clear the operation list and thus performing an undo command will cause an error since no undos are available, but if there was an error with the new_tracker command, then the undo will show us the old status. For the new_phase command, undoing will remove a phase instead if there were no errors. For the remove_phase command, undoing will add a phase instead if there were no errors, and has a way of saving the removed phase once removed so that we can add the phase again when the undo command is run. For the add_container command, the undo command will instead remove the container if there were no errors. For the move_container command, in order to undo, we need to keep track of the moved container before moving it, that way when undo is run, we can add the container back to the phase it was moved from, and remove it from the phase it was moved to. Finally for the remove_container command, in order to undo we need to keep track of both the phase it was removed from and the container removed, that way when we undo we can add the container back to the phase it was removed from.

5. Significant Contracts (Correctness)

The significant contracts are mainly in the TRACKER class itself, and there is also one for the PHASE class.

The TRACKER class has the following contracts:

invariant:
- **phase_radiation_not_greater_than_max_allowable**: across phases as phase all phase.item.radiation <= maximum_phase_radiation end
- **container_radiation_not_greater_than_max_allowable**: across containers as container all container.item.radiation_amount <= maximum_container_radiation end
- **container_amount_in_phase_not_greater_than_max_allowable**: across phases as phase all phase.item.containers.count <= phase.item.max_containers end
- **phase_does_not_handle_unexpected_material**: do_all_phases_handle_correct_material
- **container_resides_in_one_phase_only**: across containers as container all is_container_in_one_phase (container.item.container_id) end

The first one checks all phases to see if they exceed the maximum allowable radiation for a phase in the tracker system. The second one checks all containers to see if any container exceeds the maximum allowable radiation amount for a container in the tracker system. The third one checks all phases and checks in each phase if the container capacity in the phase has been exceeded. The fourth one checks all phases, and in each phase it checks the containers to see if the correct material is used in the container for the phase. Finally, the fifth one checks that there is no duplicate container in the system, meaning that the same container can't be found in two different phases. With all these checks, we satisfy the third design goal for our tracker system which deals with the safety of our system.

The PHASE class has the following contract:

Invariant:
- **max_containers_not_exceeded**: containers.count <= max_containers

This invariant ensures that the container capacity in the phase is not exceeded. This check is done here as well as in the contract for the TRACKER class, so it's not entirely necessary but it still is useful to have to ensure safety within the system.

6. Summary of Testing Procedures

Test file	**Description**	**Passed**
at1.txt	Testing when starting off creating a new tracker with errors, then immediately undoing and redoing. Testing simple errors that occur with making a phase, testing creating and moving containers, and vigorously testing undo and redo for all commands.	✓
at2.txt	Lots of testing for undo and redo for the different commands.	✓
at3.txt	Testing a few errors that can occur with the new_tracker command, new_phase command, and the move_container command.	✓
at4.txt	Testing some errors that can occur with creating a new container and moving a container.	✓
at5.txt	Testing tons of possible errors that can occur and seeing if the program can handle all these possible scenarios correctly.	✓

7. Appendix (Contract view of all classes, i.e. their *specification*)

TRACKER:

```
note
	description: "A default business model."
	author: "Jackie Wang"
	date: "$Date$"
	revision: "$Revision$"

class interface
	TRACKER

create {TRACKER_ACCESS}
	make

feature

	out: STRING_8
			-- New string containing terse printable representation
			-- of current object

feature -- Attributes

	containers: LINKED_LIST [MATERIAL_CONTAINER]

	current_status: STRING_8

	error_at_start: BOOLEAN

	error_status: STATUS

	has_redo_occurred: BOOLEAN

	has_undo_occurred: BOOLEAN

	has_undo_redo_occurred: BOOLEAN

	maximum_container_radiation: VALUE
			-- checks

	maximum_phase_radiation: VALUE

	no_new_tracker: BOOLEAN

	operation_list: LINKED_LIST [OPERATION]

	phases: LINKED_LIST [PHASE]

	redo_unavailable: BOOLEAN

	sorted_containers: LINKED_LIST [MATERIAL_CONTAINER]
```

sorted_phases: LINKED_LIST [PHASE]

state_number: INTEGER_32

undo_unavailable: BOOLEAN

feature -- Commands

add_container (container: MATERIAL_CONTAINER; pid: STRING_8)

add_container_element (container: MATERIAL_CONTAINER; p: PHASE)

clear_operation_list

increase_state_number

remove_containers_element (cid: STRING_8)

remove_phases_element (pid: STRING_8)

set_current_status (s: STRING_8)

set_maximum_container_radiation (mcr: VALUE)

set_maximum_phase_radiation (mpr: VALUE)

feature -- Constructor

make

feature -- Model Commands

move_container (container_id: STRING_8; phase_id_one: STRING_8; phase_id_two: STRING_8)

new_container (container_id: STRING_8; container_tuple: TUPLE [material_id: INTEGER_32; radiation_amount: VALUE]; phase_id: STRING_8)

new_phase (phase_id: STRING_8; phase_name: STRING_8; max_containers: INTEGER_32; materials_list: ARRAY [INTEGER_32])

new_tracker (max_phase_radiation: VALUE; max_container_radiation: VALUE)

redo

remove_container (container_id: STRING_8)

remove_phase (phase_id: STRING_8)

undo

feature -- Queries

do_all_phases_handle_correct_material: BOOLEAN

```
        get_container (cid: STRING_8): MATERIAL_CONTAINER

        get_current_status: STRING_8

        get_phase (pid: STRING_8): PHASE

        get_phase_id (cid: STRING_8): STRING_8

        get_phase_with_cid (cid: STRING_8): detachable PHASE

        get_state_number: INTEGER_32

        is_container_in_one_phase (cid: STRING_8): BOOLEAN
                require
                        material_with_cid_exists: is_material_exists (cid)

        is_container_in_phase (cid: STRING_8; pid: STRING_8): BOOLEAN

        is_container_present (cid: STRING_8): BOOLEAN

        is_container_radiation_exceeded (radiation: VALUE; pid: STRING_8): BOOLEAN

        is_material_exists (cid: STRING_8): BOOLEAN

        is_material_present (pid: STRING_8; mid: INTEGER_32): BOOLEAN

        is_material_present_in_phase (cid: STRING_8; pid: STRING_8): BOOLEAN

        is_phase_full (pid: STRING_8): BOOLEAN

        is_phase_present (pid: STRING_8): BOOLEAN

        is_phase_radiation_exceeded (cid: STRING_8; pid: STRING_8): BOOLEAN

        using_tracker: BOOLEAN

feature -- Sorting Commands

        clear_containers

        clear_phases

        clear_sorted_containers

        clear_sorted_phases

        print_containers: STRING_8

        print_phases: STRING_8

        sort_containers

        sort_phases
```

```
feature -- model operations

	default_update
			-- Perform update to the model state.

	reset
			-- Reset model state.

invariant
	phase_radiation_not_greater_than_max_allowable: across
			phases as phase
		all
			phase.item.radiation <= maximum_phase_radiation
		end
	container_radiation_not_greater_than_max_allowable: across
			containers as container
		all
			container.item.radiation_amount <= maximum_container_radiation
		end
	container_amount_in_phase_not_greater_than_max_allowable: across
			phases as phase
		all
			phase.item.containers.count <= phase.item.max_containers
		end
	phase_does_not_handle_unexpected_material: do_all_phases_handle_correct_material
	container_resides_in_one_phase_only: across
			containers as container
		all
			is_container_in_one_phase (container.item.container_id)
		end

end -- class TRACKER
```

NEW_TRACKER_COMMAND:

```
note
	description: "Summary description for {NEW_TRACKER_COMMAND}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	NEW_TRACKER_COMMAND

create
	make

feature -- Attributes

	is_empty: BOOLEAN

	max_container_radiation: VALUE
```

```
        max_phase_radiation: VALUE

        old_max_container_radiation: VALUE

        old_max_phase_radiation: VALUE

        old_status: STRING_8

        tracker: TRACKER

feature -- Constructor

        execute

        make (trkr: TRACKER; mpr: VALUE; mcr: VALUE)

        redo

        undo

end -- class NEW_TRACKER_COMMAND
```

NEW_PHASE_COMMAND:

```
note
        description: "Summary description for {NEW_PHASE_COMMAND}."
        author: ""
        date: "$Date$"
        revision: "$Revision$"

class interface
        NEW_PHASE_COMMAND

create
        make

feature -- Attributes

        is_empty: BOOLEAN

        materials_list: ARRAY [INTEGER_32]

        max_containers: INTEGER_32

        old_status: STRING_8

        phase_id: STRING_8

        phase_name: STRING_8

        tracker: TRACKER

feature -- Constructor
```

```
	execute

	make (trkr: TRACKER; pid: STRING_8; p_name: STRING_8; max: INTEGER_32; mat_list: ARRAY
[INTEGER_32])

	redo

	undo

end -- class NEW_PHASE_COMMAND
```

REMOVE_PHASE_COMMAND:

```
note
	description: "Summary description for {REMOVE_PHASE_COMMAND}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	REMOVE_PHASE_COMMAND

create
	make

feature -- Attributes

	is_empty: BOOLEAN

	old_phase: detachable PHASE

	old_status: STRING_8

	phase_id: STRING_8

	removed_phase: detachable PHASE

	tracker: TRACKER

feature -- Constructor

	execute

	make (trkr: TRACKER; pid: STRING_8)

	redo

	undo

end -- class REMOVE_PHASE_COMMAND
```

NEW_CONTAINER_COMMAND:

```
note
	description: "Summary description for {NEW_CONTAINER_COMMAND}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	NEW_CONTAINER_COMMAND

create
	make

feature -- Attributes

	container_id: STRING_8

	container_tuple: TUPLE [m_id: INTEGER_32; rad: VALUE]

	is_empty: BOOLEAN

	material_id: INTEGER_32

	old_status: STRING_8

	phase_id: STRING_8

	radiation_amount: VALUE

	tracker: TRACKER

feature -- Constructor

	execute

	make (trkr: TRACKER; cid: STRING_8; con_t: TUPLE [mid: INTEGER_32; radiation: VALUE]; pid:
STRING_8)

	redo

	undo

end -- class NEW_CONTAINER_COMMAND
```

MOVE_CONTAINER_COMMAND:

```
note
	description: "Summary description for {MOVE_CONTAINER_COMMAND}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	MOVE_CONTAINER_COMMAND

create
	make

feature -- Attributes

	container_id: STRING_8

	is_empty: BOOLEAN

	moved_container: detachable MATERIAL_CONTAINER

	old_status: STRING_8

	phase_id_one: STRING_8

	phase_id_two: STRING_8

	tracker: TRACKER

feature -- Constructor

	execute

	make (trkr: TRACKER; cid: STRING_8; pid_one: STRING_8; pid_two: STRING_8)

	redo

	undo

end -- class MOVE_CONTAINER_COMMAND
```

REMOVE_CONTAINER_COMMAND:

```
note
	description: "Summary description for {REMOVE_CONTAINER_COMMAND}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	REMOVE_CONTAINER_COMMAND

create
	make

feature -- Attributes

	container_id: STRING_8

	is_empty: BOOLEAN

	old_status: STRING_8

	phase: detachable PHASE

	removed_container: detachable MATERIAL_CONTAINER

	tracker: TRACKER

feature -- Constructor

	execute

	make (trkr: TRACKER; cid: STRING_8)

	redo

	undo

end -- class REMOVE_CONTAINER_COMMAND
```

PHASE:

```
note
	description: "Summary description for {PHASE}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	PHASE

create
	make

feature -- Attributes

	containers: LINKED_LIST [MATERIAL_CONTAINER]

	materials: ARRAY [STRING_8]

	materials_list: ARRAY [INTEGER_32]

	mats: STRING_8

	max_containers: INTEGER_32

	phase_id: STRING_8

	phase_name: STRING_8

feature -- Commands

	add_container (new_container: MATERIAL_CONTAINER)

	clear_containers

	remove_container (cid: STRING_8)

feature -- Constructor

	make (pid: STRING_8; pn: STRING_8; max: INTEGER_32; mat_list: ARRAY [INTEGER_32])

feature -- Queries

	full: BOOLEAN

	is_container_present (cid: STRING_8): BOOLEAN

	radiation: VALUE

invariant
	max_containers_not_exceeded: containers.count <= max_containers
```

```
end -- class PHASE
```

MATERIAL_CONTAINER:

```
note
	description: "Summary description for {MATERIAL_CONTAINER}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	MATERIAL_CONTAINER

create
	make

feature -- Attributes

	container_id: STRING_8

	material: STRING_8

	material_id: INTEGER_32

	radiation_amount: VALUE

feature -- Command

	set_material (s: STRING_8)

feature -- Constructor

	make (cid: STRING_8; mid: INTEGER_32; radiation: VALUE)

end -- class MATERIAL_CONTAINER
```

STATUS:

```
note
	description: "Summary description for {STATUS}."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	STATUS

create
	make

feature

	make

feature -- Attributes

	E1: STRING_8 = "e1: current tracker is in use"

	E10: STRING_8 = "e10: this container identifier already in tracker"

	E11: STRING_8 = "e11: this container will exceed phase capacity"

	E12: STRING_8 = "e12: this container will exceed phase safe radiation"

	E13: STRING_8 = "e13: phase does not expect this container material"

	E14: STRING_8 = "e14: container radiation capacity exceeded"

	E15: STRING_8 = "e15: this container identifier not in tracker"

	E16: STRING_8 = "e16: source and target phase identifier must be different"

	E17: STRING_8 = "e17: this container identifier is not in the source phase"

	E18: STRING_8 = "e18: this container radiation must not be negative"

	E19: STRING_8 = "e19: there is no more to undo"

	E2: STRING_8 = "e2: max phase radiation must be non-negative value"

	E20: STRING_8 = "e20: there is no more to redo"

	E3: STRING_8 = "e3: max container radiation must be non-negative value"

	E4: STRING_8 = "e4: max container must not be more than max phase radiation"

	E5: STRING_8 = "e5: identifiers/names must start with A-Z, a-z or 0..9"

	E6: STRING_8 = "e6: phase identifier already exists"
```

```
        E7: STRING_8 = "e7: phase capacity must be a positive integer"

        E8: STRING_8 = "e8: there must be at least one expected material for this phase"

        E9: STRING_8 = "e9: phase identifier not in the system"

        is_empty: BOOLEAN

        Ok_status: STRING_8 = "ok"

feature -- Commands

        get_is_empty: BOOLEAN

        set_is_empty (e: BOOLEAN)

end -- class STATUS
```