

CPD3314 Assignment No.10

Build the Following Assignment and Submit to Dropbox on or before Apr. 16th, 2015

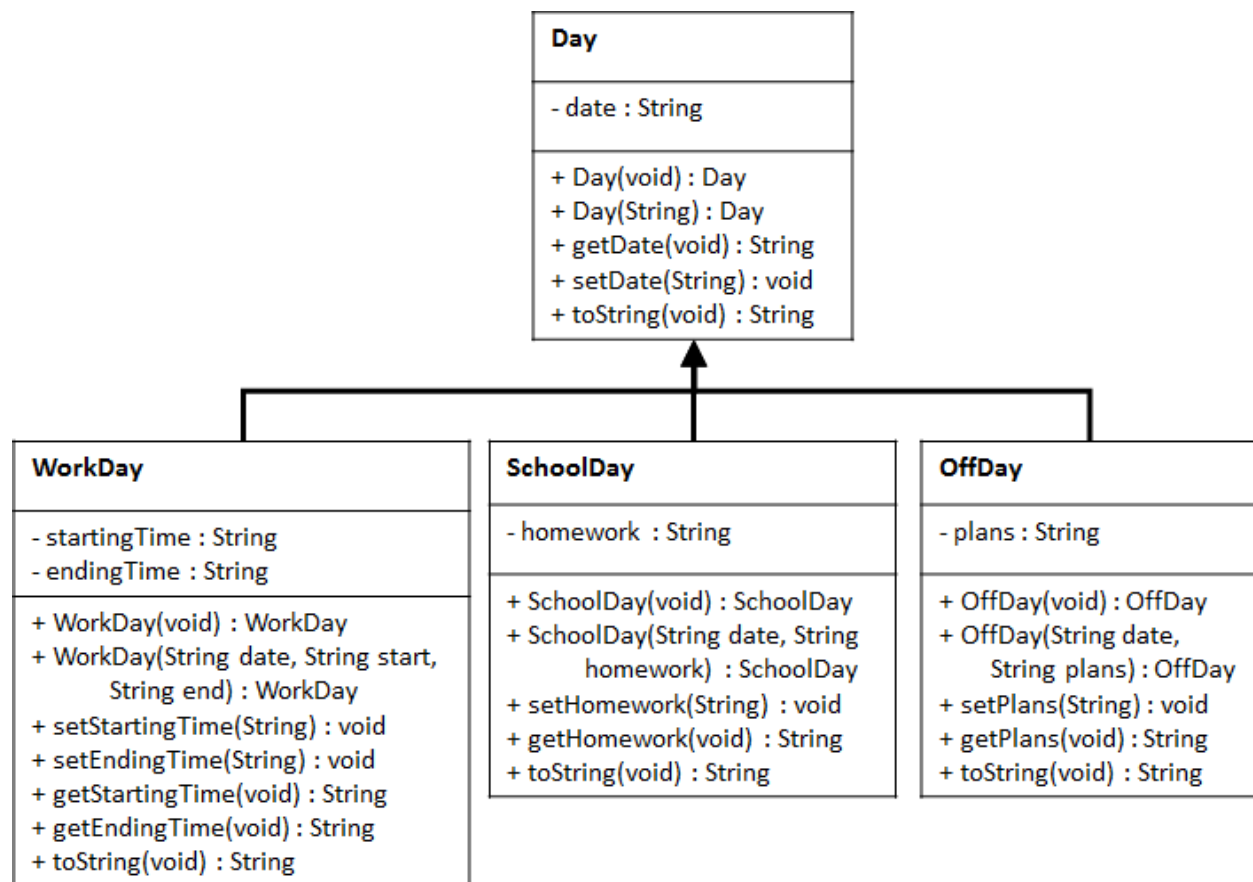
This task requires you to use many principles covered in the course, but highlights inheritance, exception-handling, and graphical user interfaces. Specifically, you will need to create a hierarchical data model that you can update using an error-proof GUI.

You will be creating a weekly scheduling program.

The interface will show seven days. Each day will either be a work-day, a school-day, or a day-off. All of them will be days, but the interface will show different data depending on the type of day.

Data Model

Create a **Day** class, and extend it into **WorkDay**, **School Day**, and **OffDay**. You will be creating an array of seven **Day** references to represent a Week. See below for UML diagrams for the various **Days**.



Starting files have been provided to outline and test this data model. They are available via D2L and GitHub. They are not exhaustive tests, but allow us to ensure you are on the right path before building your User Interface. Please confirm you are passing the data model tests before worrying about the User Interface classes and arrangement.

The `toString()` methods should give the following outputs:

- `Day toString()` → Just outputs date.
 - Eg – `new Day("4").toString()` → "4"
- `WorkDay toString()` → "%s : Work : %s-%s", date, start, end
 - Eg – `new WorkDay("4", "9:00", "5:00").toString()` → "4 : Work : 9:00-5:00"
- `School Day toString()` → "%s : School : %s", date, homework
 - Eg – `new School Day("4", "Java Assign 10").toString()` → "4 : School : Java Assign 10"
- `OffDay toString()` → "%s : Off : %s", date, plans
 - Eg- `new OffDay("4", "Watch Football").toString()` → "4 : Off : Watch Football"

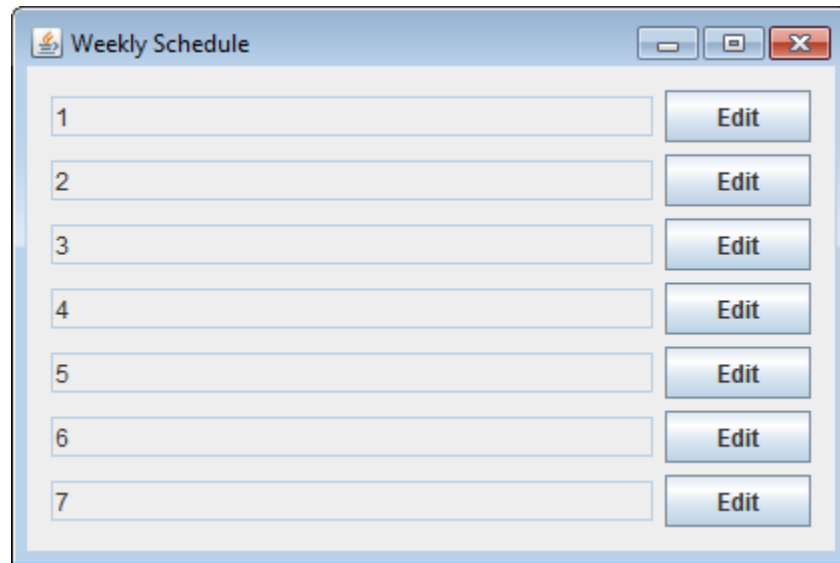
Bonus – Convert Times to Integers – Storing Hours/Minutes Separate

- For an added challenge (worth a 100% bonus) convert the data storage of hours/minutes to be as integers. The interface does not need to change, only the implementation of `WorkDay`. Instead of having `startingTime` as a `String`, store it as `startHour` and `startMin`, two integers. The trick here is verifying a time is a time, and converting something like "9:00" into 9 and 0, or "9am" into 9 and 0 without causing an absolute crash, and still giving the user feedback if they try to save something like "I:KL".

Controller (Main Class)

- Create a **private static** array of 7 `Day` objects
- Create a **private static** reference to a `Schedule Frame` object
- Create **static public** methods for:
 - `setDay(int, Day)` to replace a `Day` in the array, and call an update on the `Schedule Frame`
 - `getDay(int)` to retrieve a `Day` from the array
 - `editDay(int)` to Spawn an `Edit JFrame` based on an index and one of the `Day` objects
- In the **main** method:
 - Initialize the `Day` array to have regular `Day` objects, with dates 1-7
 - Create a new `Schedule Frame` instance
 - Call the `setVisible(true)` method on the `Schedule Frame` object

User Interface



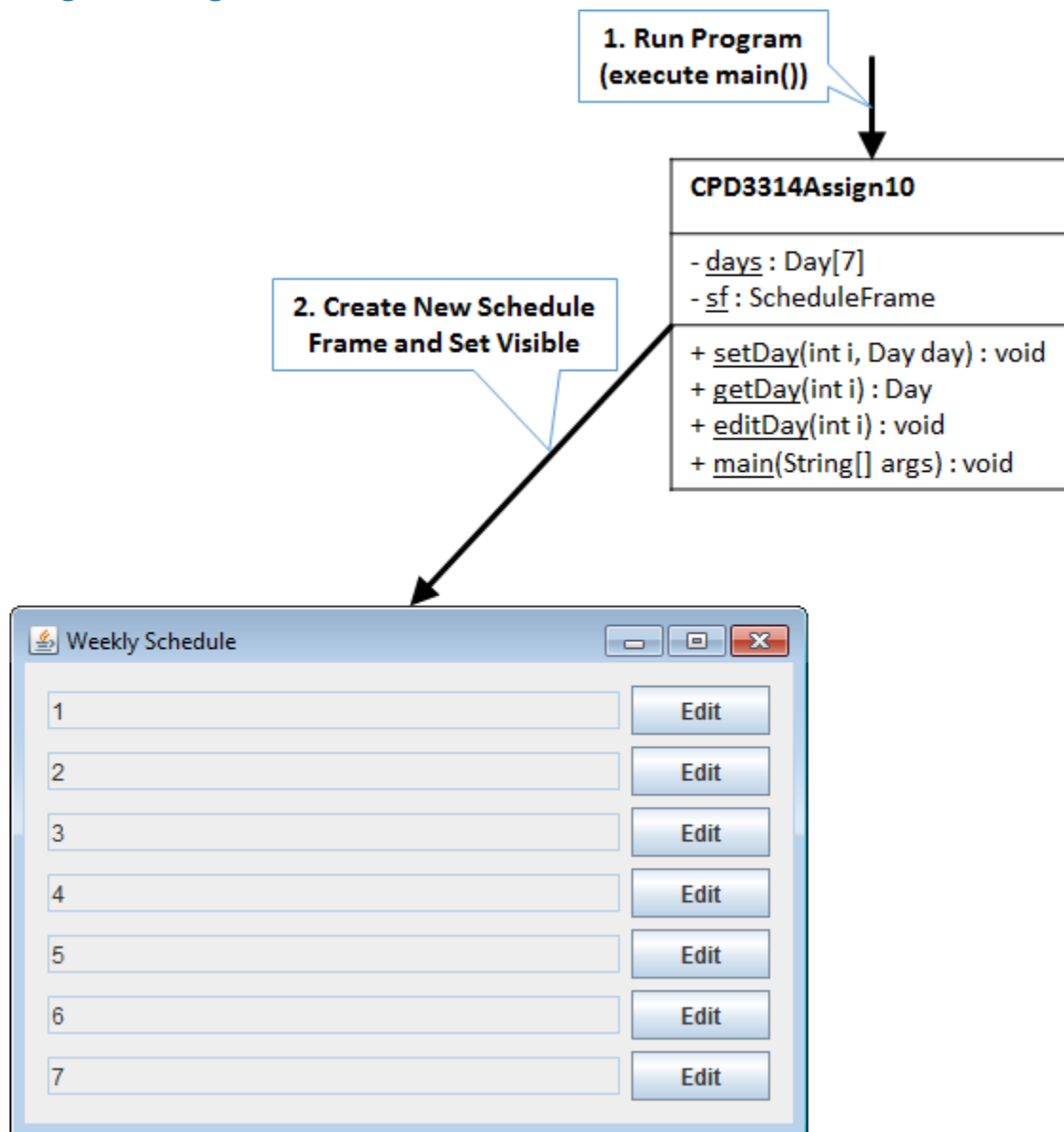
- **JFrame**
 - Title: "Weekly Schedule"
 - Seven **JTextFields**
 - Text: <blank>
 - Not Editable
 - Seven **JButtons**
 - Text: Edit
- Behaviours:
 - Must provide **public void refreshDays()** – Calls **getDay(int)** from the Controller to update the **JTextFields** with the **toString()** method of each **Day**
 - Button Methods – Call back to the Controller's **editDay(int)** method to spawn a new Edit Frame

The screenshot shows a Java Swing dialog box titled "Edit Day". It has a standard window header with minimize, maximize, and close buttons. The dialog contains the following elements:

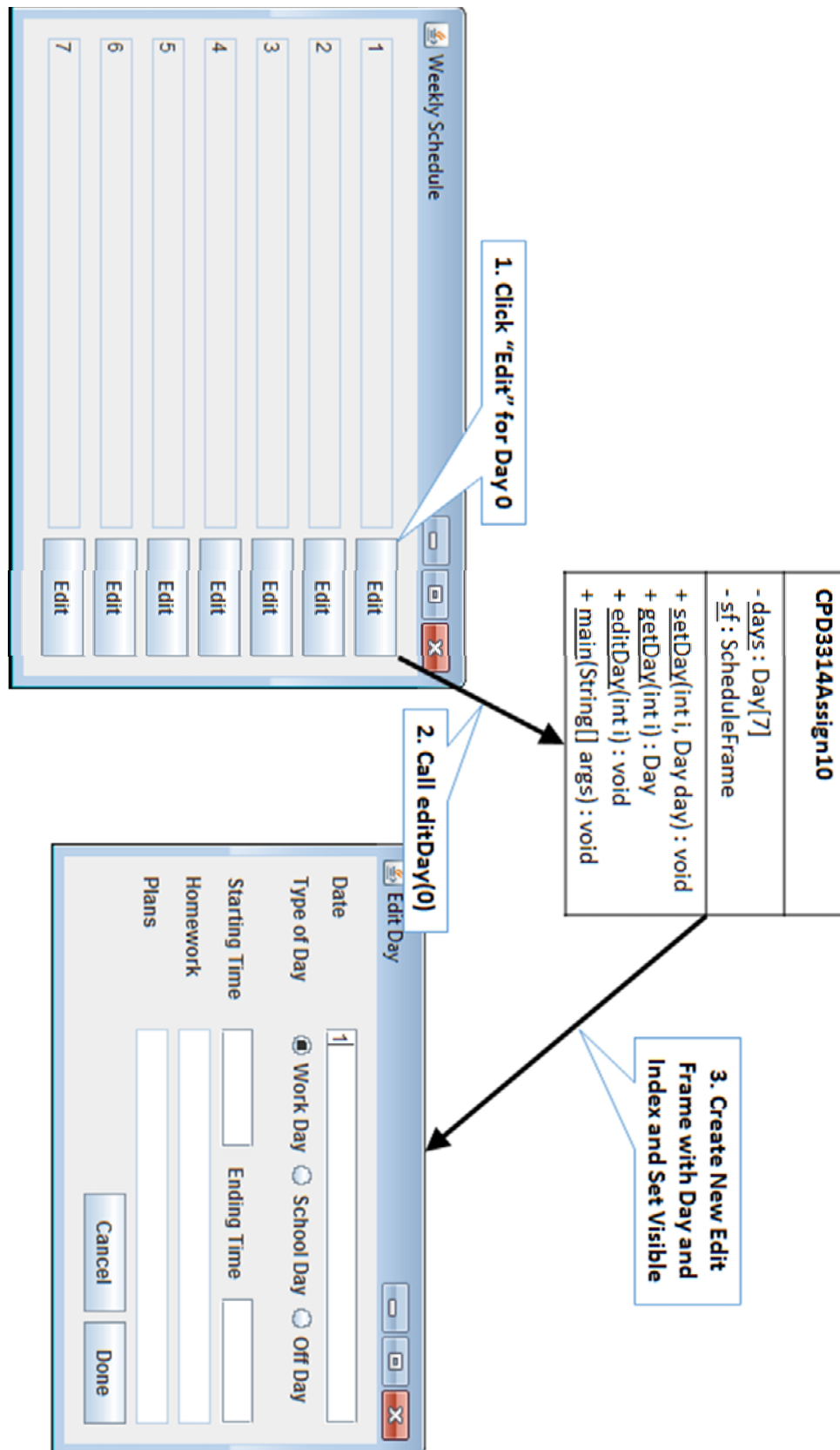
- Date:** A text field containing the value "1".
- Type of Day:** Three radio buttons labeled "Work Day", "School Day", and "Off Day". The "Work Day" radio button is selected.
- Starting Time:** A text field containing "9:00".
- Ending Time:** A text field containing "5:00".
- Homework:** An empty text field.
- Plans:** An empty text field.
- Buttons:** Two buttons at the bottom right labeled "Cancel" and "Done".

- **JFrame**
 - Title "Edit Day"
 - Six **JLabel**s
 - Text: Date, Type of Day, Starting Time, Ending Time, Homework, Plans
 - Five **JTextField**s
 - For Date, Starting Time, Ending Time, Homework, Plans
 - Three **JRadioButton**s
 - Common Button Group
 - Text: Work Day, School Day, Off Day
 - Two **JButtons**
 - Text: Cancel, Done
- **Behaviours:**
 - Default starts as a Work Day
 - If Work Day is selected, enable Start and End Time Text Fields, but disable Homework and Plans Text Fields
 - If School Day is selected, enable Homework Text Fields, but disable Start, End and Plans Text Fields
 - If Off Day is selected, enable Plans Text Fields, but disable Start, End and Homework Text Fields
 - When Cancel is pressed, **setVisible(false)** and **dispose()** the Frame
 - When Done is pressed, create a new appropriate **Day** sub-object that matches the user's selections, and use the Controller's **setDay(int, Day)** method to update the array of **Days**, then **setVisible(false)** and **dispose()** the Frame

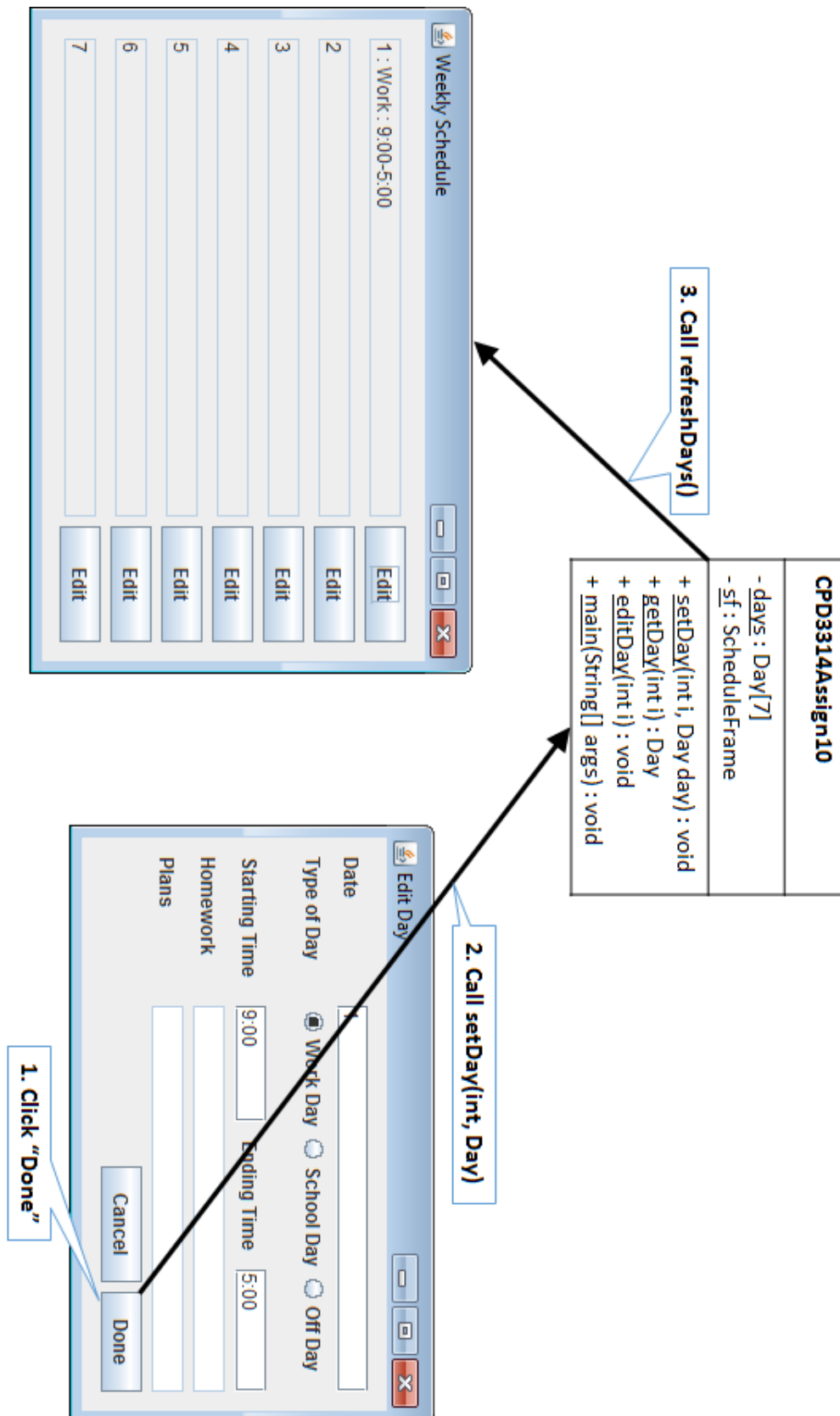
Starting the Program



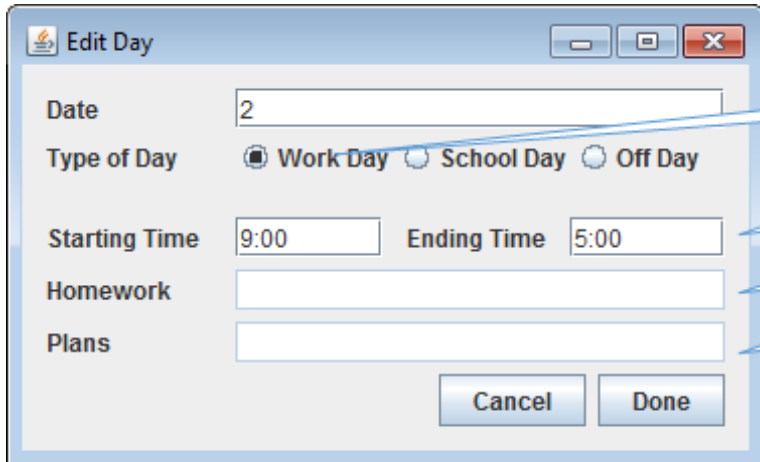
Selecting "Edit" for a Day



Selecting “Done” on an Edit Frame



Selecting Options in an Edit Frame



Edit Day

Date: 2

Type of Day: ☒ Work Day ☐ School Day ☐ Off Day

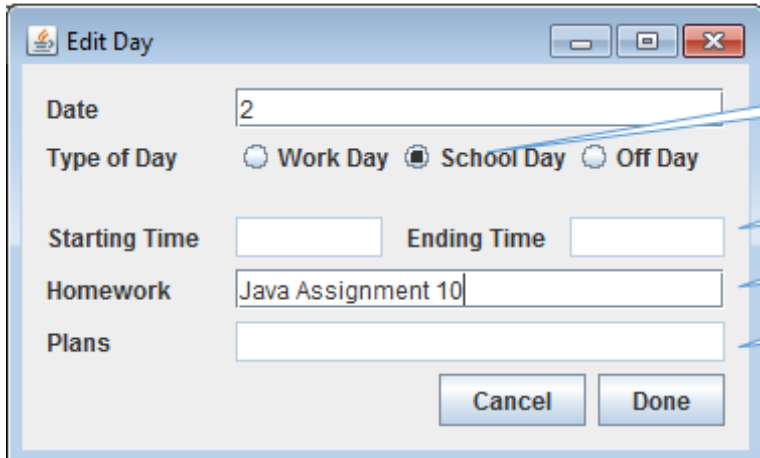
Starting Time: 9:00 Ending Time: 5:00

Homework:

Plans:

Cancel Done

- 1. Work Day Selected
- 2. Start and End Enabled
- 3. Homework Disabled
- 4. Plans Disabled



Edit Day

Date: 2

Type of Day: ☐ Work Day ☒ School Day ☐ Off Day

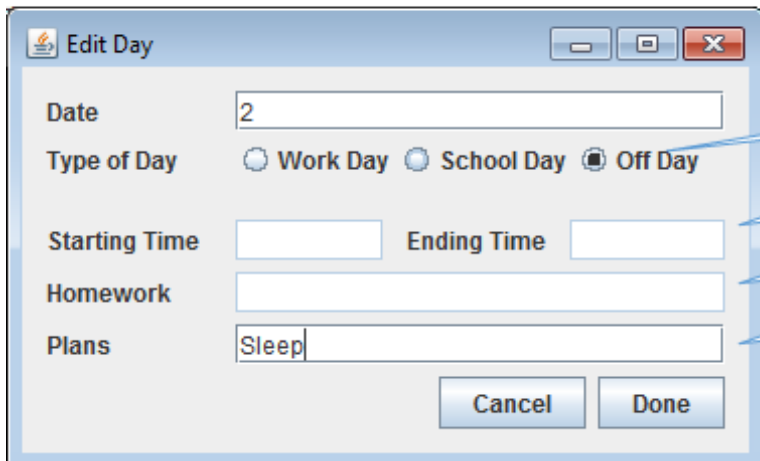
Starting Time: Ending Time:

Homework: Java Assignment 10

Plans:

Cancel Done

- 1. School Day Selected
- 2. Start and End Disabled
- 3. Homework Enabled
- 4. Plans Disabled



Edit Day

Date: 2

Type of Day: ☐ Work Day ☐ School Day ☒ Off Day

Starting Time: Ending Time:

Homework:

Plans: Sleep

Cancel Done

- 1. Off Day Selected
- 2. Start and End Disabled
- 3. Homework Disabled
- 4. Plans Enabled