Recitation 1- Data Structures

Adithya Iyer, 26 Jan 2024

Recitation Logistics

- Recitation : Bobst LL138, 9:30-10:45AM
- Everything we do in recitations will be uploaded here. (Link on Brightspace)
- Recitations will mostly be :
  - Getting to solve 2-3 Qs : Approach
  - In later recitations : actually coding it up (efficiently)
  - Rough Idea - get you to be better at interviews?

About me

- PLEASE call me Adi (~~Prof Adi~~ / ~~Adithya~~ / ~~Professor~~)
- How I got here?
  - Undergrad in Physics/Mat Sci @ IIT Bombay
  - Management Consulting @ McKinsey & Co.
  - Computer Vision Research @ NYU-X (Prof Saining Xie)
- Email : ai2257@nyu.edu
- OH : Monday - 3-4 pm WWH 705

Today ?

- Quick Recap of things with examples
- **Getting Eclipse Working**
- An exercise on using JAVA OOP to solve complex problems

The Problem, the Slides and the Code will all be up on the GitHub repository.

Recap :

- https://github.com/adithyaiyer1999/data-structures-recitations-nyu2024/blob/main/Recitation01/Recitation01.md

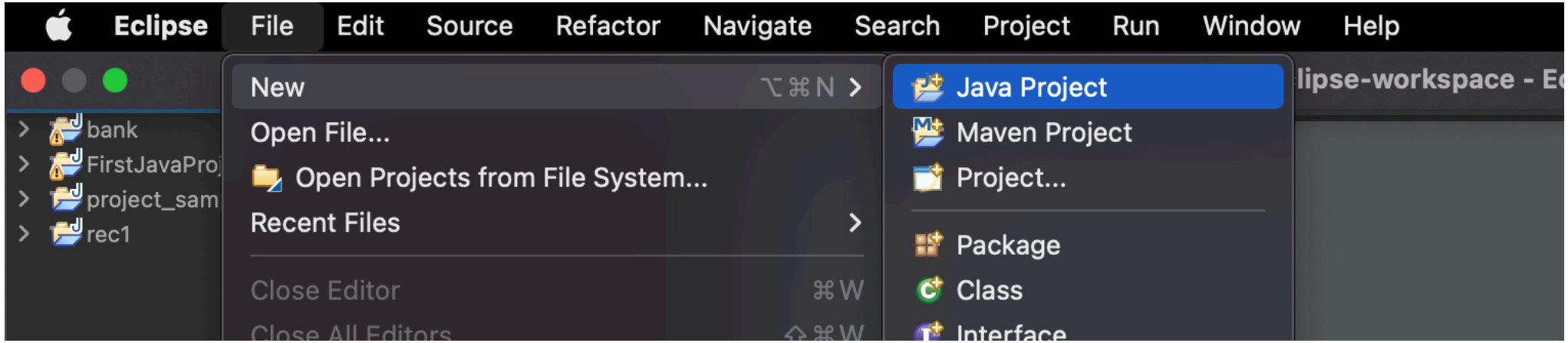
Setting up Eclipse - Level Easy

- https://github.com/adithyaiyer1999/data-structures-recitations-nyu2024/blob/main/Recitation01/setup-eclipse.md

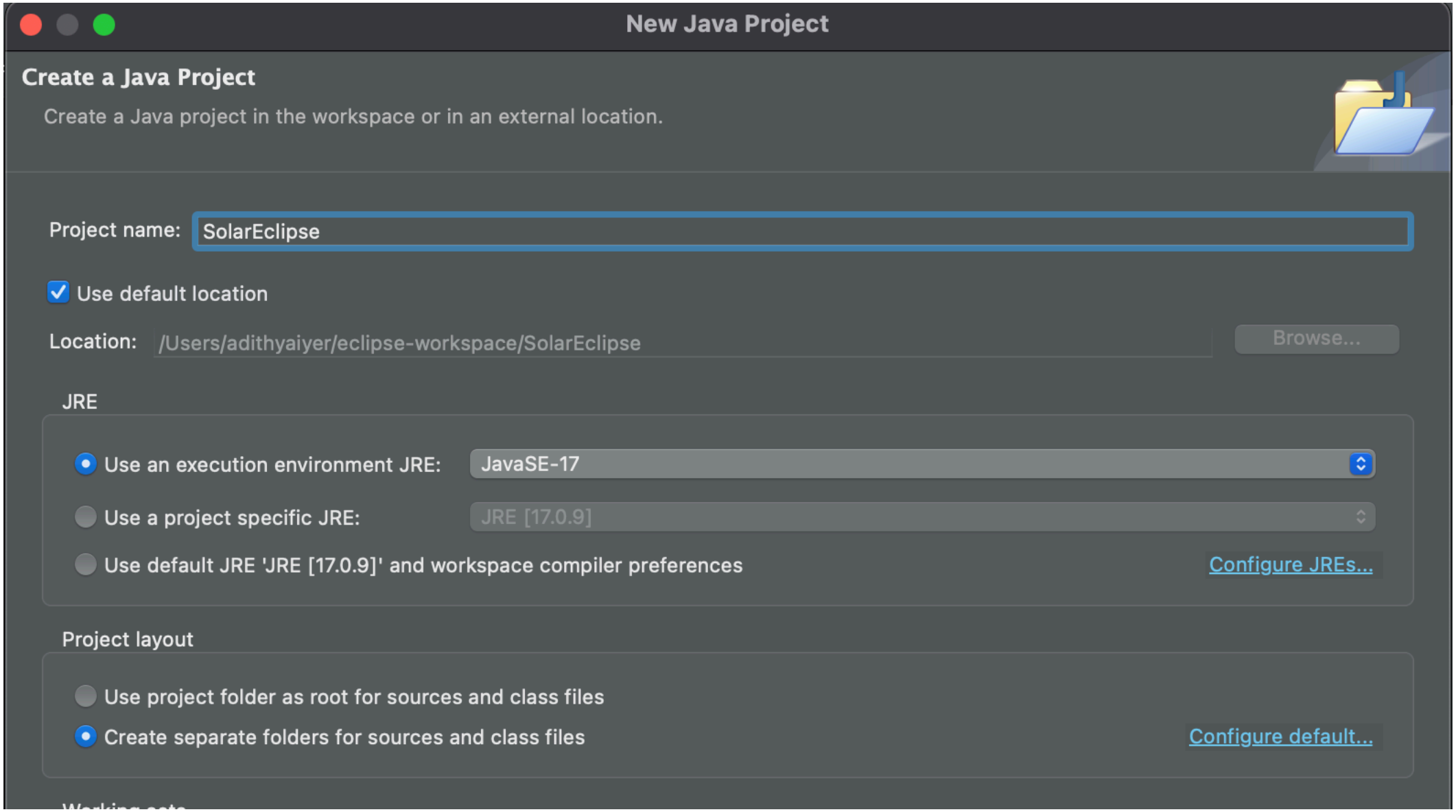
Setting up Eclipse - Level Hard

- We’ll setup folders, and call an object from a different folder
- If this works in Eclipse - lucky you
- If this does not work in Eclipse -
  - Agree that Eclipse is a bad product :(
  - Reach out to me after class

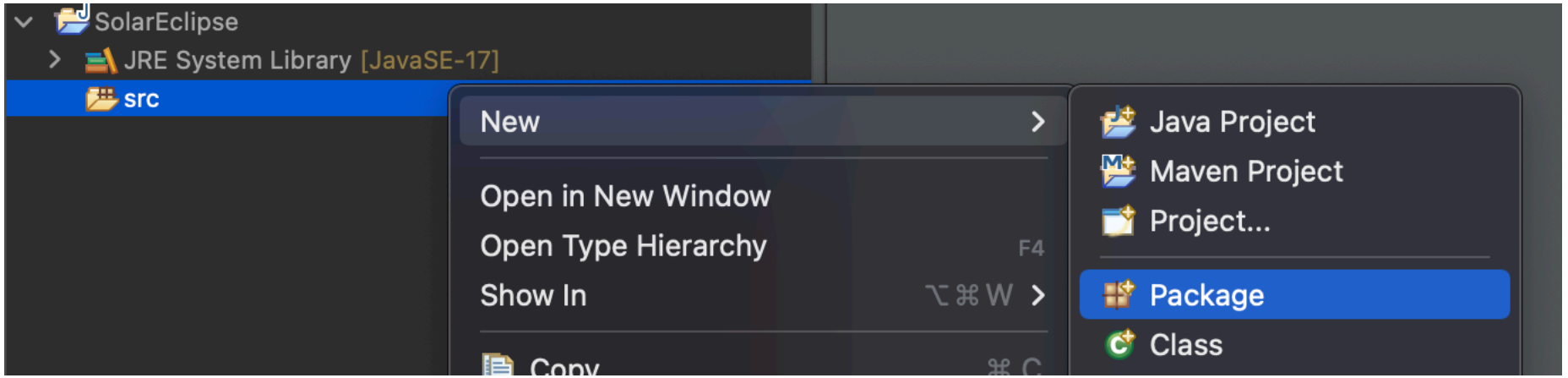
Step 1


Eclipse
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
bank
FirstJavaProj
project_sam
rec1
New
Open File...
Open Projects from File System...
Recent Files
Close Editor
Java Project
Maven Project
Project...
Package
Class
Interface

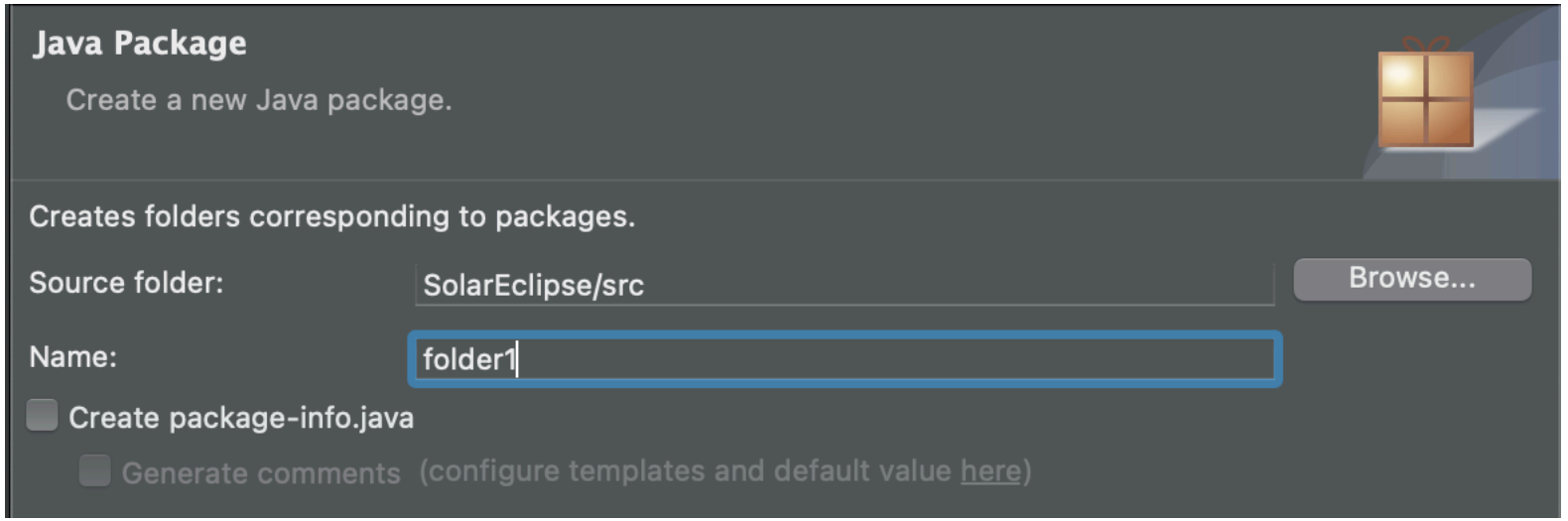
Step2


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name:
SolarEclipse
Use default location
Location:
/Users/adithyaiyer/eclipse-workspace/SolarEclipse
Browse...
JRE
Use an execution environment JRE:
JavaSE-17
Use a project specific JRE:
JRE [17.0.9]
Use default JRE 'JRE [17.0.9]' and workspace compiler preferences
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...

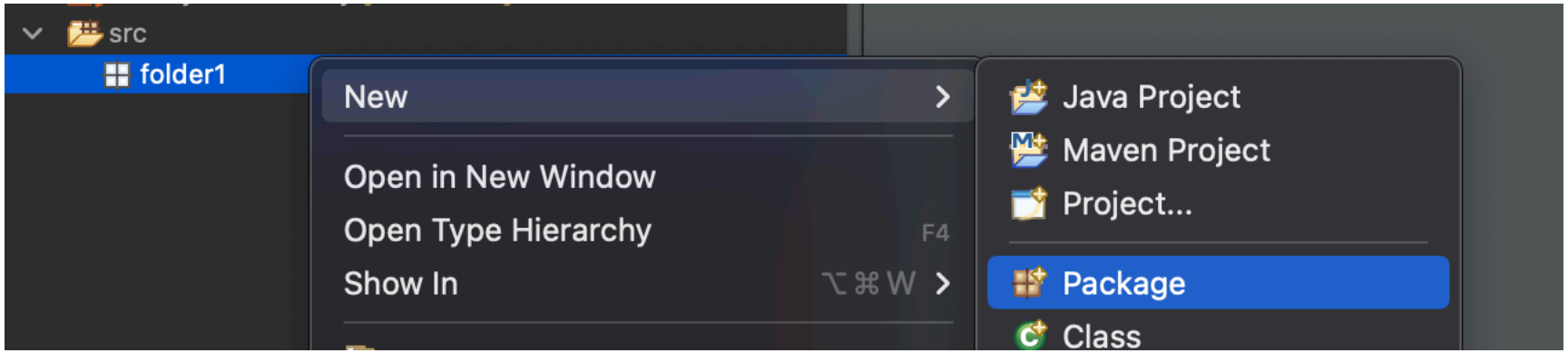
Step3


SolarEclipse
JRE System Library [JavaSE-17]
src
New
Open in New Window
Open Type Hierarchy
F4
Show In
⌥⌘W
Copy
⌘C
Java Project
Maven Project
Project...
Package
Class

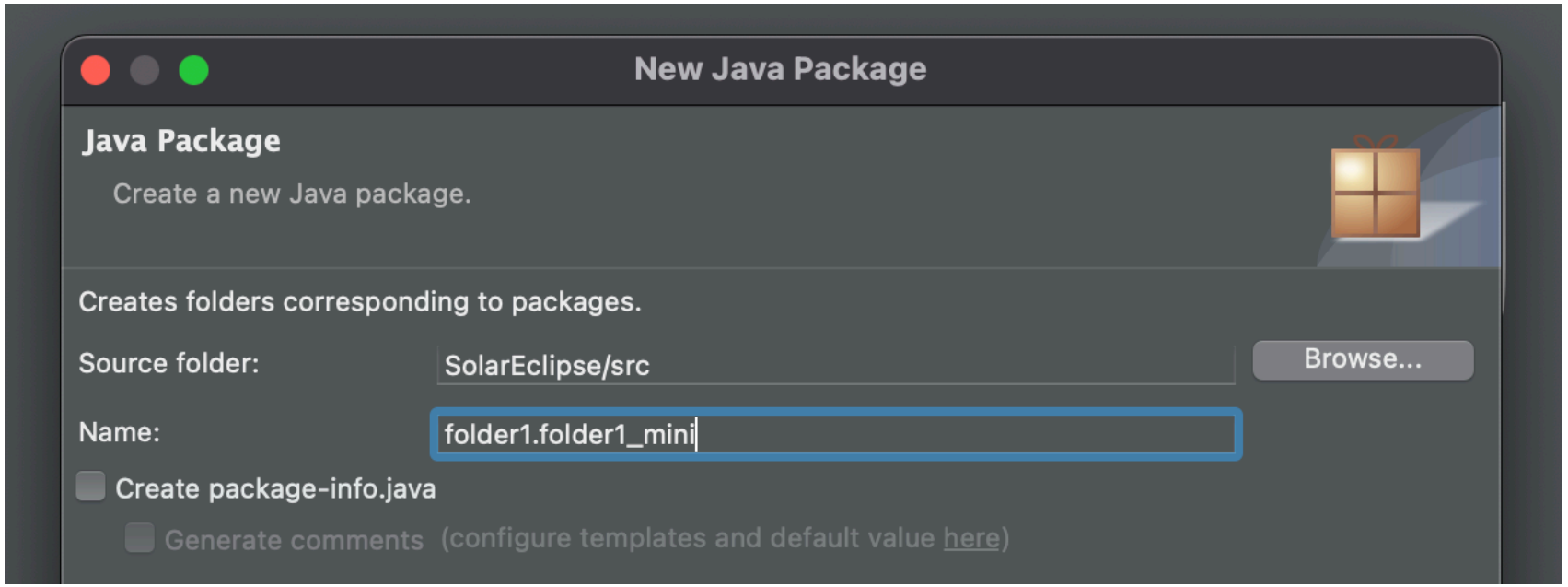
Step4


Java Package
Create a new Java package.
Creates folders corresponding to packages.
Source folder:
SolarEclipse/src
Browse...
Name:
folder1
Create package-info.java
Generate comments (configure templates and default value here)

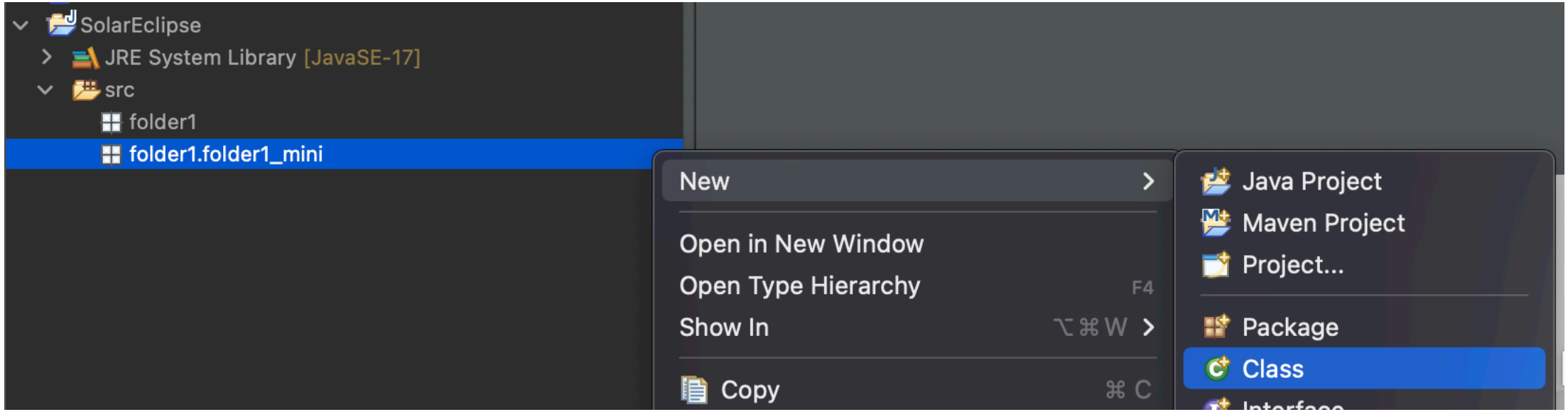
Step 5


src
folder1
New
Open in New Window
Open Type Hierarchy
F4
Show In
⌥⌘W
Java Project
Maven Project
Project...
Package
Class

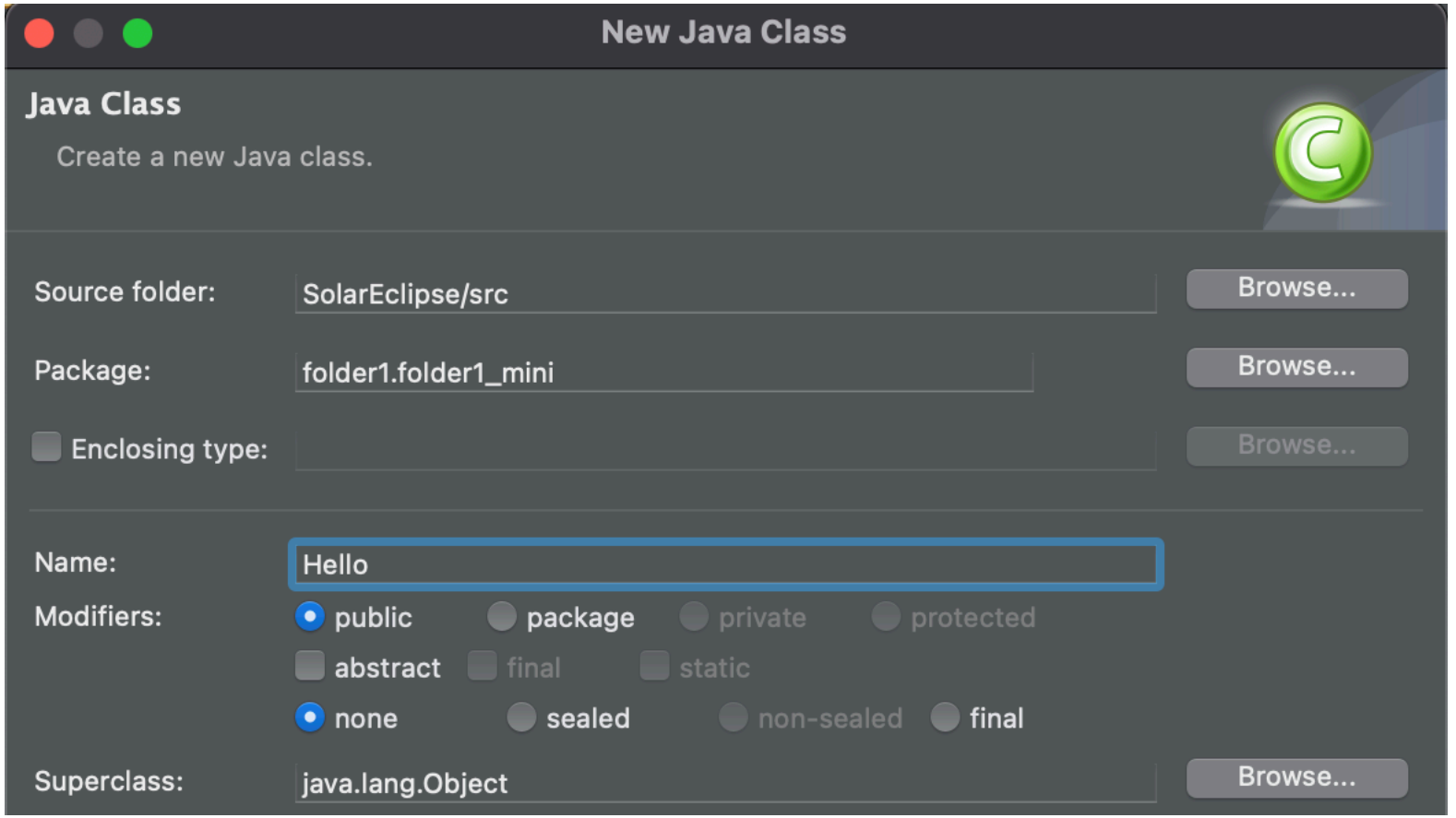
Step 6


New Java Package
Java Package
Create a new Java package.
Creates folders corresponding to packages.
Source folder:
SolarEclipse/src
Browse...
Name:
folder1.folder1_mini
Create package-info.java
Generate comments (configure templates and default value here)

Step 7


SolarEclipse
JRE System Library [JavaSE-17]
src
folder1
folder1.folder1_mini
New
Open in New Window
Open Type Hierarchy
F4
Show In
⌥⌘W
Copy
⌘C
Java Project
Maven Project
Project...
Package
Class

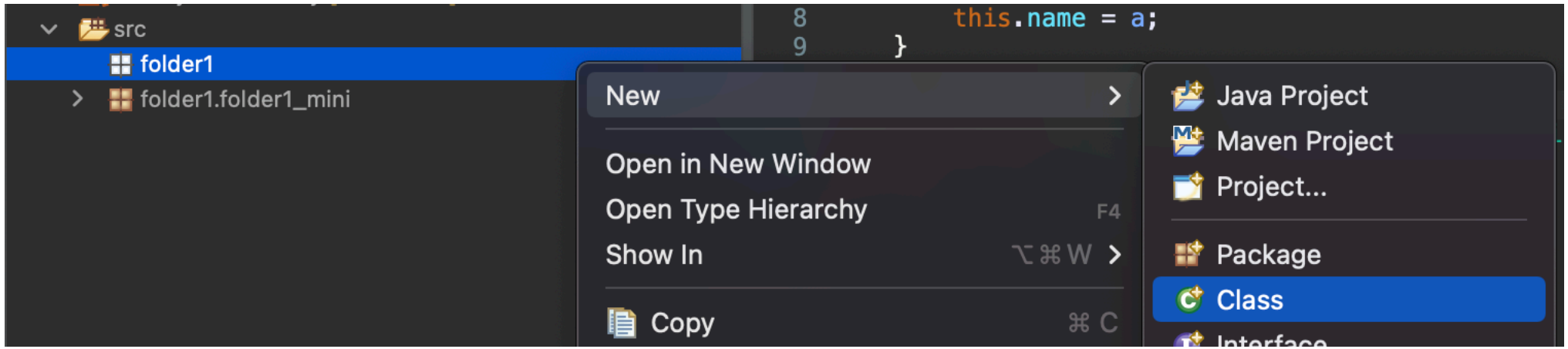
Step 8


New Java Class
Java Class
Create a new Java class.
Source folder:
SolarEclipse/src
Browse...
Package:
folder1.folder1_mini
Browse...
Enclosing type:
Browse...
Name:
Hello
Modifiers:
public
package
private
protected
abstract
final
static
none
sealed
non-sealed
final
Superclass:
java.lang.Object
Browse...

Step 9


Hello.java
1 package folder1.folder1_mini;
2
3 public class Hello {
4
5 private String name;
6
7 public Hello(String a) {
8 this.name = a;
9 }
10
11 public void sayHi() {
12 System.out.println("Hello from folder1_mini !");
13 }
14 }
15

Step 10

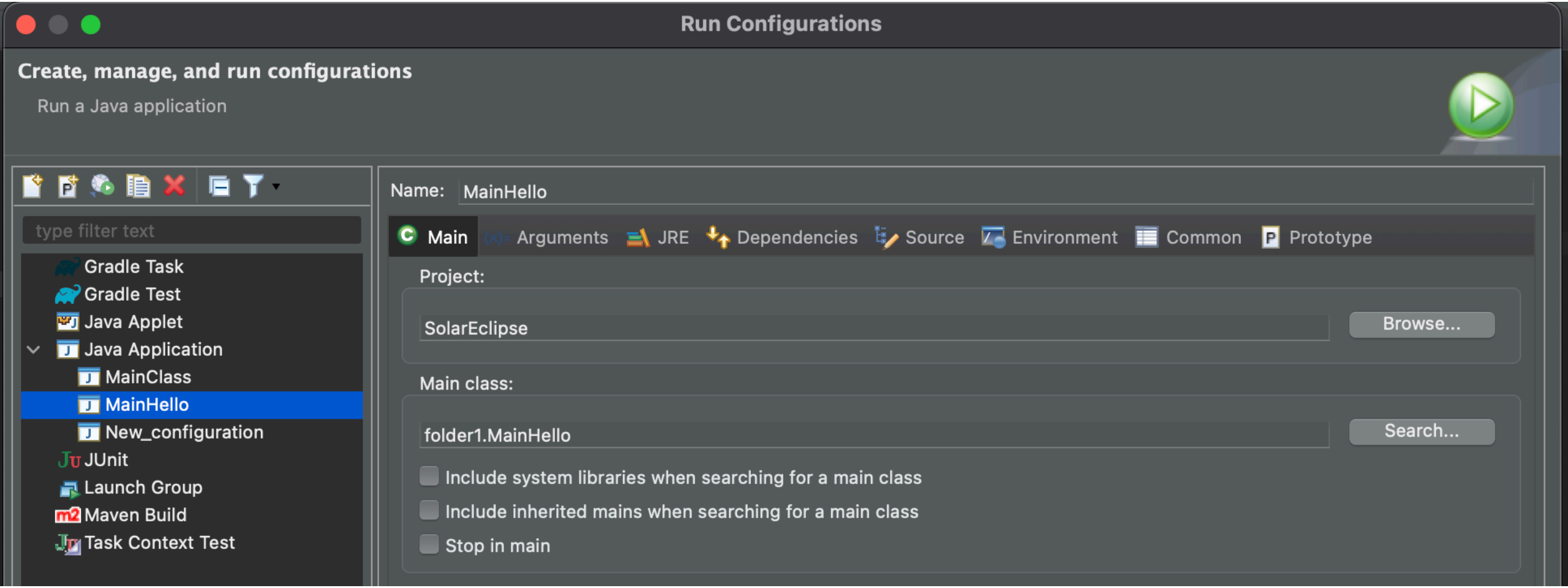

src
folder1
folder1.folder1_mini
8 this.name = a;
9 }
New
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Java Project
Maven Project
Project...
Package
Class

Step 11


Hello.java
MainHello.java
1 package folder1;
2 import folder1.folder1_mini.*;
3
4 public class MainHello {
5 public static void main(String[] args) {
6 Hello yellow = new Hello("random string");
7 yellow.sayHi();
8 }
9
10 }
11

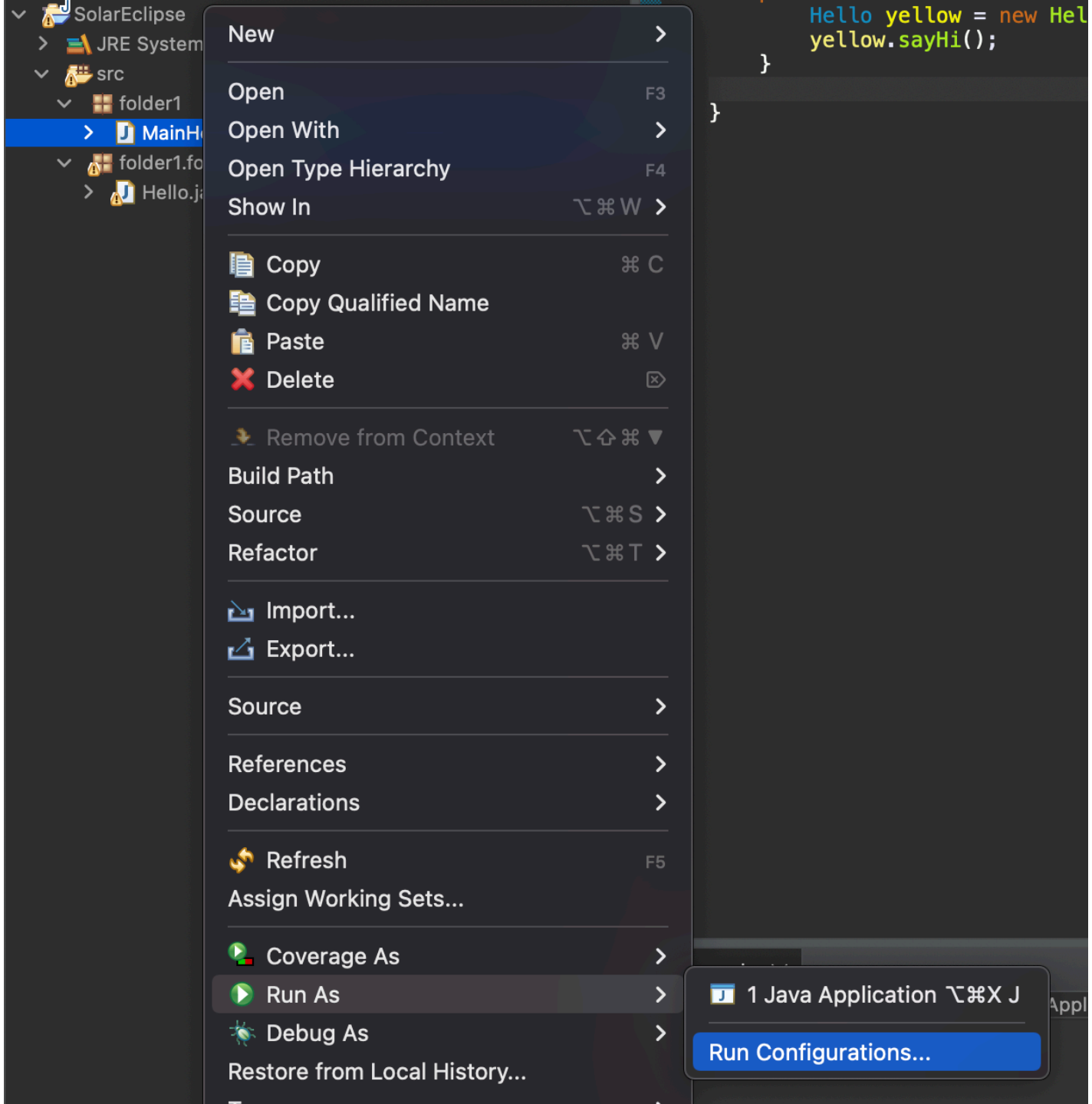
Very very very important to not mess this up.

Step 12



Step 13

vvVVvvvvvvvvv
Vvvvvvvimmpp

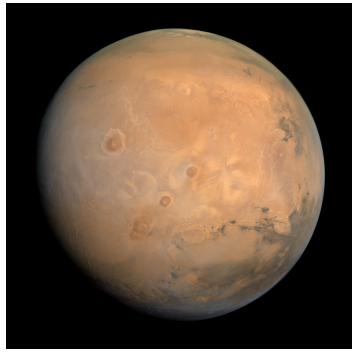
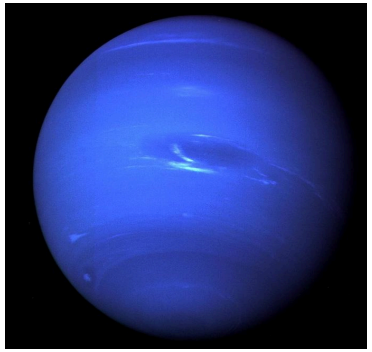


Step 14 - Success

Moving on to OOP ?

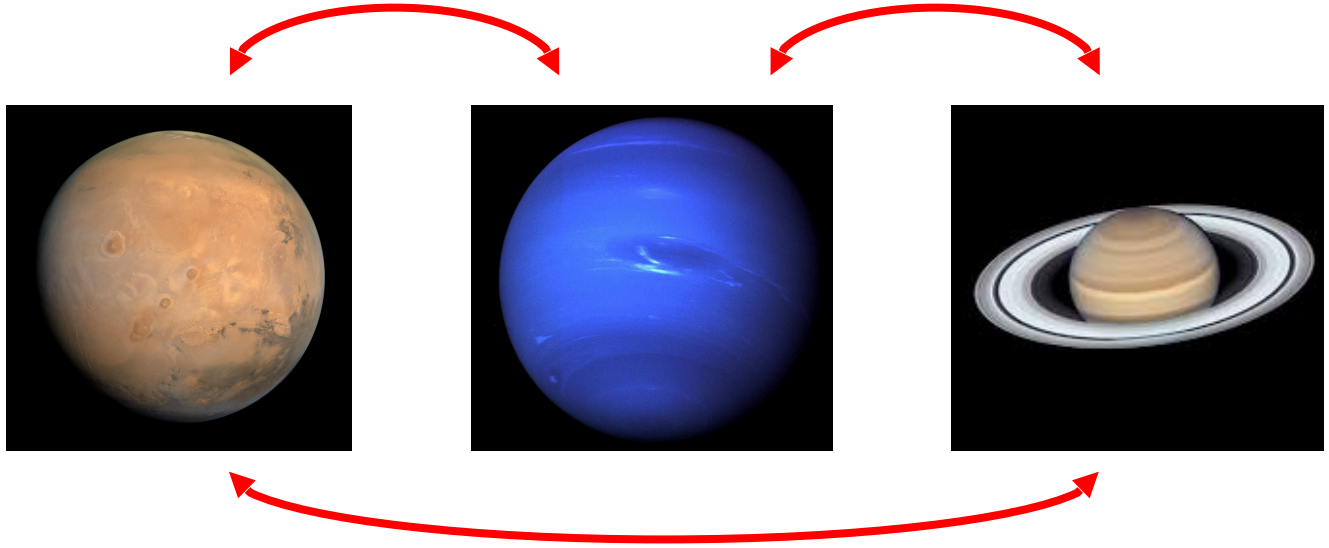
Scenario

- People from Mars, Neptune, and Saturn come to buy and sell items.
- They all have different currencies.

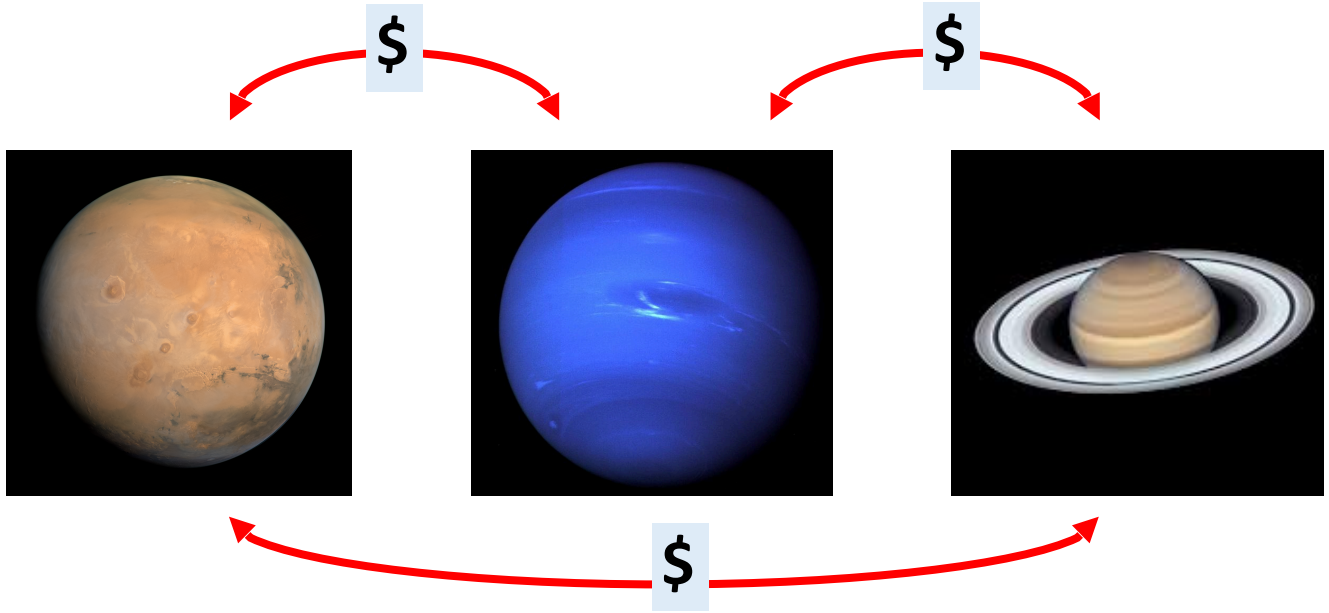

Scenario

- People from Mars, Neptune, and Saturn come to buy and sell items.
- They all have different currencies.
- Earth dollars are universally accepted, so we will use it as the intermediary to help **exchange** between their own currencies (each planet also charges a fee).



Scenario

- People from Mars, Neptune, and Saturn come to buy and sell items.
- They all have different currencies.
- **Earth dollars** are universally accepted, so we will use it as the intermediary to help **exchange** between their own currencies (each planet also charges a fee).



Sample run

Input

```
Currency mars = new Mars(100.00);
Currency neptune = new Neptune(100.00);
Currency saturn = new Saturn(100.00);

System.out.println("<-- Exchanges -->");

mars.exchange(saturn, 25.00);
neptune.exchange(saturn, 10.00);
saturn.exchange(mars, 122.00);
saturn.exchange(mars, 121.00);
```

What we really want to achieve.

Output

```
<-- Exchanges -->
Converting from MarsMoney to SaturnSilver and initiating transfer…
25.00 MarsMoney = 19.23 EarthDollars = 16.73 SaturnSilver
Mars exchange fee is 2.50 MarsMoney
Mars has a total of 72.50 MarsMoney
Saturn has a total of 116.73 SaturnSilver

Converting from NeptuneNuggets to SaturnSilver and initiating transfer…
10.00 NeptuneNuggets = 5.00 EarthDollars = 4.35 SaturnSilver
Neptune exchange fee is 5.00 NeptuneNuggets
Neptune has a total of 85.00 NeptuneNuggets
Saturn has a total of 121.08 SaturnSilver

Uh oh - Saturn only has an available balance of 121.08, which is less than
122.00!

Converting from SaturnSilver to MarsMoney and initiating transfer…
121.00 SaturnSilver = 139.08 EarthDollars = 180.80 MarsMoney
Saturn exchange fee is 0.00 SaturnSilver
Saturn has a total of 0.08 SaturnSilver
Mars has a total of 253.30 MarsMoney
```

What are the common actions?

- The common action (method) among all planet currencies was the **exchange**.
- These shared actions can be abstracted into **interfaces**.

What are the shared attributes?

- The currencies of different planets had common properties such as a **name** and **total funds**.
- This suggests the need for a parent or **abstract class**.

What properties are unique to each planet?

- Each planet currency has **their own exchange rates** and **fees**.
- Each planet currency class should **extend (inherit) and implement the specifics.**
  - toEarthDollars
  - fromEarthDollars
  - getTransactionFee

Define the Interface

```
package currencyexchange.utility;

public interface Exchangeable {
    double EARTH_DOLLAR_TO_MARS_MONEY = 1.3;
    double EARTH_DOLLAR_TO_SATURN_SILVER = 0.87;
    double EARTH_DOLLAR_TO_NEPTUNE_NUGGET = 2.0;

    void exchange(Exchangeable other, double amount);

}
```

- Define constants for exchange rates

Define the Interface

```
package currencyexchange.utility;

public interface Exchangeable {
    double EARTH_DOLLAR_TO_MARS_MONEY = 1.3;
    double EARTH_DOLLAR_TO_SATURN_SILVER = 0.87;
    double EARTH_DOLLAR_TO_NEPTUNE_NUGGET = 2.0;

    void exchange(Exchangeable other, double amount);

}
```

- Define constants for exchange rates
- Define the abstract method, **exchange**

Abstract Parent Class

```java
package currencyexchange.currency;

import currencyexchange.utility.Exchangeable;

public abstract class Currency implements Exchangeable{
    protected String currencyName;
    protected double totalFunds;

    public abstract double toEarthDollars(double amount);
    public abstract double fromEarthDollars(double EarthDollars);

    protected abstract double getTransactionFee(double amount);

    @Override
    public void exchange(Exchangeable other, double amount) {
```

- Define attributes: **currencyName** and **totalFunds**

Sub classes - Mars

```java
package currencyexchange.currency;

public class Mars extends Currency {

    public Mars(double initialFunds) {
        this.currencyName = "MarsMoney";
        this.totalFunds = initialFunds;
    }

    @Override
    public double toEarthDollars(double amount) {
        return amount / EARTH_DOLLAR_TO_MARS_MONEY;
    }

    @Override
    public double fromEarthDollars(double EarthDollars) {
        double marsMoney = EarthDollars * EARTH_DOLLAR_TO_MARS_MONI
        this.totalFunds += marsMoney;
        return marsMoney;

    }

    @Override
    protected double getTransactionFee(double amount) {
        return amount * 0.1;
    }
}
```

- **Constructor** sets the currencyName and initial totalFunds.

Sub classes - Mars

```java
package currencyexchange.currency;

public class Mars extends Currency {

    public Mars(double initialFunds) {
        this.currencyName = "MarsMoney";
        this.totalFunds = initialFunds;
    }

    @Override
    public double toEarthDollars(double amount) {
        return amount / EARTH_DOLLAR_TO_MARS_MONEY;
    }

    @Override
    public double fromEarthDollars(double EarthDollars) {
        double marsMoney = EarthDollars * EARTH_DOLLAR_TO_MARS_MONI
        this.totalFunds += marsMoney;
        return marsMoney;

    }

    @Override
    protected double getTransactionFee(double amount) {
        return amount * 0.1;
    }
}
```

- **Override the abstract methods**: toEarthDollars / fromEarthDollars / getTrasactionFee

Main class

```java
package currencyexchange;

import currencyexchange.currency.*;

public class Main {
    public static void main(String[] args) {
        Currency mars = new Mars(100.00);
        Currency neptune = new Neptune(100.00);
        Currency saturn = new Saturn(100.00);

        System.out.println("<-- Exchanges -->");

        mars.exchange(saturn, 25.00);
        neptune.exchange(saturn, 10.00);
        saturn.exchange(mars, 122.00);
        saturn.exchange(mars, 121.00);
    }
}
```

- **Test** and evaluate your implementation.

Abstract Parent Class

```java
package currencyexchange.currency;

import currencyexchange.utility.Exchangeable;

public abstract class Currency implements Exchangeable{
    protected String currencyName;
    protected double totalFunds;

    public abstract double toEarthDollars(double amount);
    public abstract double fromEarthDollars(double EarthDollars);

    protected abstract double getTransactionFee(double amount);

    @Override
    public void exchange(Exchangeable other, double amount) {
```

- Define attributes: **currencyName** and **totalFunds**
- Define three abstract methods: **toEarthDollars**, **fromEarthDollars**, and **getTransactionFee**

Abstract Parent Class

```java
package currencyexchange.currency;

import currencyexchange.utility.Exchangeable;

public abstract class Currency implements Exchangeable{
    protected String currencyName;
    protected double totalFunds;

    public abstract double toEarthDollars(double amount);
    public abstract double fromEarthDollars(double EarthDollars);

    protected abstract double getTransactionFee(double amount);

    @Override
    public void exchange(Exchangeable other, double amount) {
```

- Define attributes: **currencyName** and **totalFunds**
- Define three abstract methods: **toEarthDollars**, **fromEarthDollars**, and **getTransactionFee**
- **Implement exchange**

exchange

```
@Override
public void exchange(Exchangeable other, double amount) {
    double transactionFee = getTransactionFee(amount);
    double AmountInEarthDollars = toEarthDollars(amount);

    if (this.totalFunds >= (amount + transactionFee)) {
        this.totalFunds -= (amount + transactionFee);
        double otherCurrencyAmount = ((Currency) other).fromEarthDollars(AmountInEarthDollars);

        System.out.println("Converting from " + this.currencyName + " to " +
                           ((Currency) other).currencyName + " and initiating transfer…");
        System.out.println(String.format("%.2f %s = %.2f EarthDollars = %.2f %s",
                                         amount, this.currencyName, AmountInEarthDollars,
                                         otherCurrencyAmount, ((Currency) other).currencyName));
        System.out.println(String.format("%s exchange fee is %.2f %s",
                                         this.currencyName, transactionFee, this.currencyName));
        System.out.println(String.format("%s has a total of %.2f %s",
                                         this.currencyName, this.totalFunds, this.currencyName));
        System.out.println(String.format("%s has a total of %.2f %s",
                                         ((Currency) other).currencyName,
                                         ((Currency) other).totalFunds,
                                         ((Currency) other).currencyName));

    } else {
        System.err.println(String.format(
                "Uh oh - %s only has an available balance of %.2f, "
                + "which is less than %.2f!",
                this.currencyName, this.totalFunds, amount + transactionFee));
    }

}
```

- **Exception handling with the control structure, if/else**: check if the current total transaction amount does not exceed the total fund currently held.

exchange

```java
@Override
public void exchange(Exchangeable other, double amount) {
    double transactionFee = getTransactionFee(amount);
    double AmountInEarthDollars = toEarthDollars(amount);

    if (this.totalFunds >= (amount + transactionFee)) {
        this.totalFunds -= (amount + transactionFee);
        double otherCurrencyAmount = ((Currency) other).fromEarthDollars(AmountInEarthDollars);

        System.out.println("Converting from " + this.currencyName + " to " +
                           ((Currency) other).currencyName + " and initiating transfer…");
        System.out.println(String.format("%.2f %s = %.2f EarthDollars = %.2f %s",
                                         amount, this.currencyName, AmountInEarthDollars,
                                         otherCurrencyAmount, ((Currency) other).currencyName));
        System.out.println(String.format("%s exchange fee is %.2f %s",
                                         this.currencyName, transactionFee, this.currencyName));
        System.out.println(String.format("%s has a total of %.2f %s",
                                         this.currencyName, this.totalFunds, this.currencyName));
        System.out.println(String.format("%s has a total of %.2f %s",
                                         ((Currency) other).currencyName,
                                         ((Currency) other).totalFunds,
                                         ((Currency) other).currencyName));

    } else {
        System.err.println(String.format(
                "Uh oh - %s only has an available balance of %.2f, "
                + "which is less than %.2f!",
                this.currencyName, this.totalFunds, amount + transactionFee));
    }

}
```

- **Perform exchange between the sender and receiver:** sender -= (amount + fee) / receiver += amount