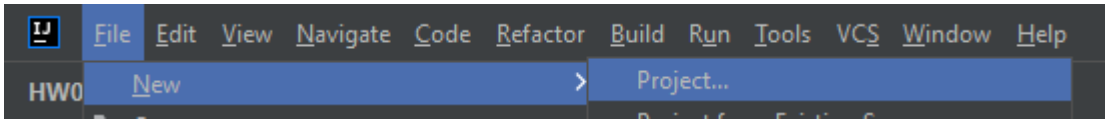
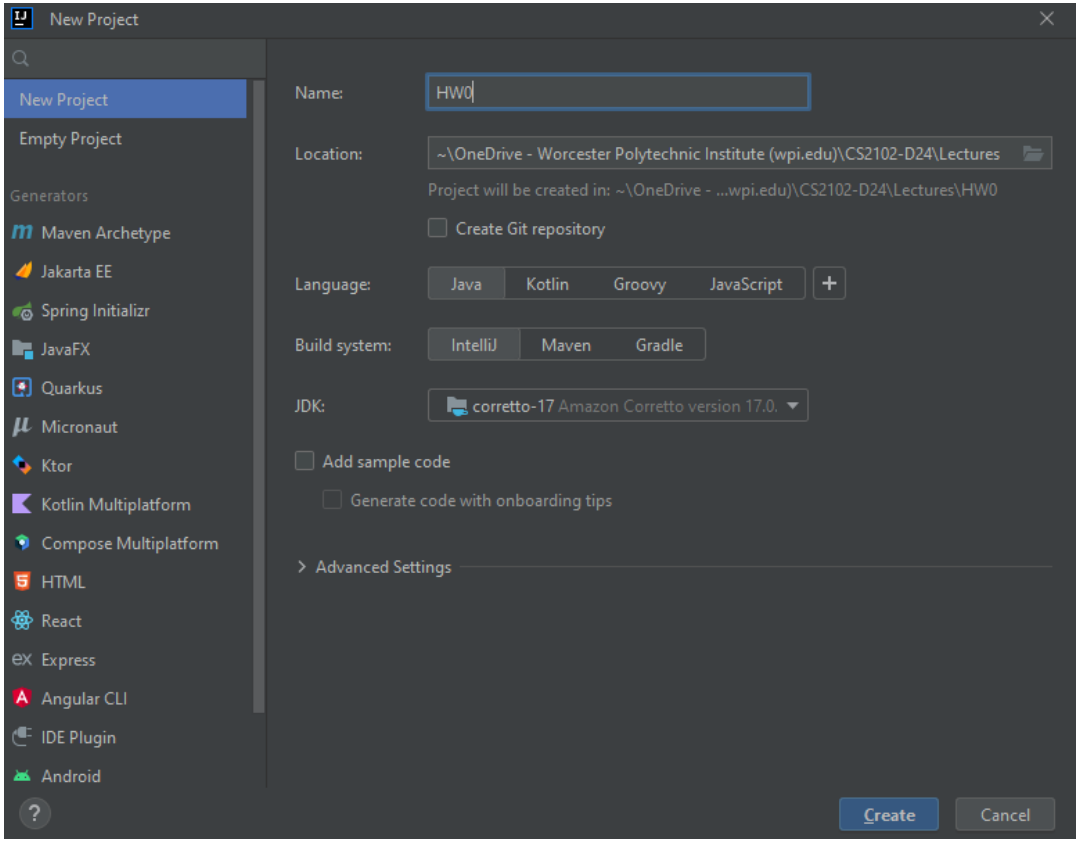
Lab 3 – TempHumid Sensors

Profs. Ahrens, Sun – B24 - CS2102

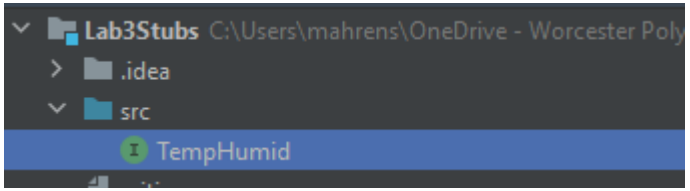
Make a new project





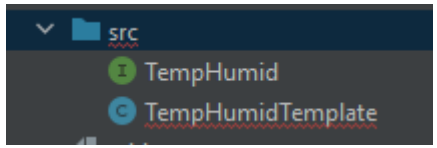
Copy TempHumid.java from the HW prompt

- **Put the interface from the homework TempHumid.java in your src/ folder**
- Define a class TempHumidTemplate .java
- Make it implement TempHumid
- Let IntelliJ generate all the method stubs
- Then make TempHumidTemplate abstract



Copy TempHumid.java from the HW prompt

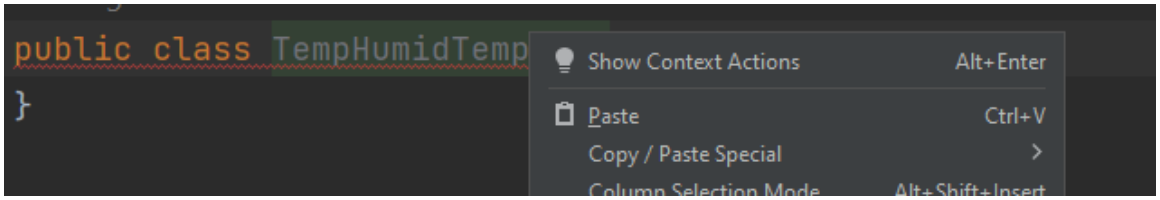
- Put the interface from the homework TempHumid.java in your src/ folder
- **Define a class TempHumidTemplate.java**
- **Make it implement TempHumid**
- Let IntelliJ generate all the method stubs
- Then make TempHumidTemplate abstract

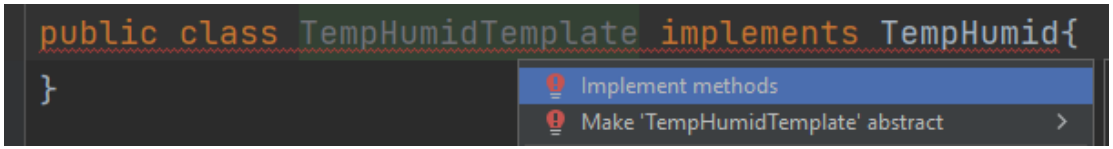


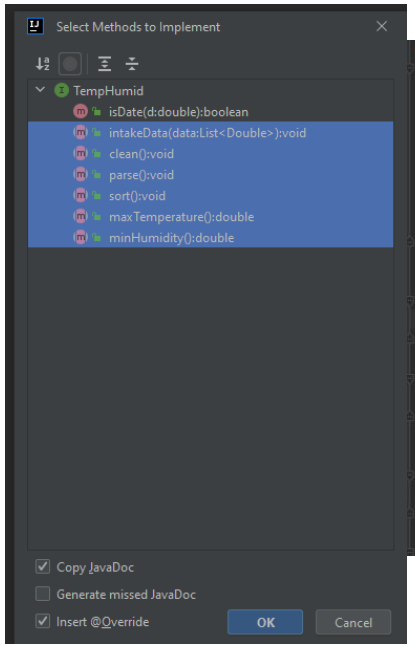

```
public class TempHumidTemplate implements TempHumid{
}
```

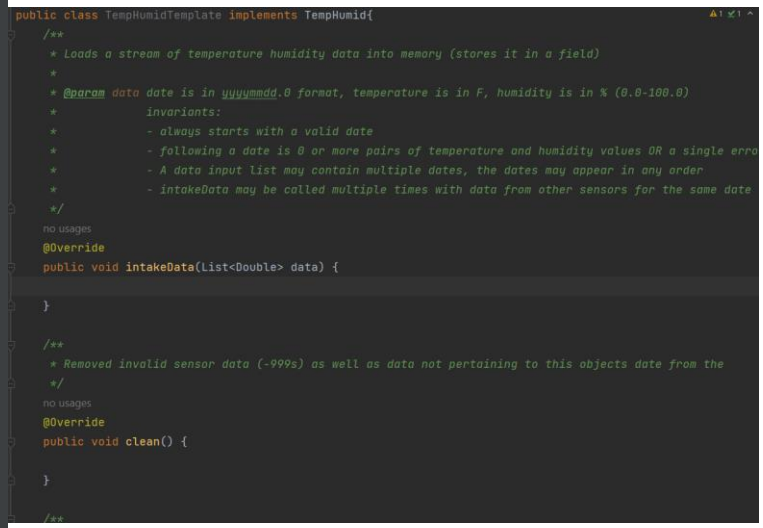
Copy TempHumid.java from the HW prompt

- Put the interface from the homework TempHumid.java in your src/ folder
- Define a class TempHumidTemplate .java
- Make it implement TempHumid
- **Let IntelliJ generate all the method stubs**
- Then make TempHumidTemplate abstract









Copy TempHumid.java from the HW prompt

- Put the interface from the homework TempHumid.java in your src/ folder
- Define a class TempHumidTemplate.java
- Make it implement TempHumid
- Let IntelliJ generate all the method stubs
- **Then make TempHumidTemplate abstract**

```
public abstract class TempHumidTemplate implements TempHumid{
```

Make three subclass stubs

- TempHumidBP
  - 0-arg constructor
  - No fields
- TempHumidRTP
  - 0-arg constructor
  - No fields
- TempHumidRTPByDate
  - This one has a 1-arg constructor
    That consumes a double date

```
public class TempHumidBP extends TempHumidTemplate{
}
```

```
public class TempHumidRTP extends TempHumidTemplate{
}
```

```
public class TempHumidRTPByDate extends TempHumidRTP{
}
```

What does input data look like for my tests?

- The input data is a list of 0 or more sequences of doubles:
- A date starting each sequence in format: yyyymmdd.0
- Followed by 0 or more pairs of temperature (F) and humidity (%)
- A pair may be replaced by an error reading (-999.0)

```
List<Double> testData = List.of( ...elements: 20240401.0,75.0,10.0,76.0,11.0,77.0,9.0,65.0,12.0,-999.0, 75.0,10.0,
                                 20240402.0 ,75.0,10.0,76.0,11.0,78.0,15.0,65.0,12.0,-999.0,75.0,10.0);
```

- You may use this data or your own

maxTemperature() for BP and RTP

- Must test with concrete subclasses only
- Cannot instantiate abstract class directly
- Should produce the same answer since there is no difference between what Batch Processing vs Real-Time Processing compute
- Only difference is in *how* they compute it

```java
@Test
public void testMaxTemp1(){
    TempHumid thd = new TempHumidBP();
    thd.intakeData(testData);
    assertEquals(???,thd.maxTemperature(),0.01);
}

@Test
public void testMaxTemp2(){
    TempHumid thd = new TempHumidRTP();
    thd.intakeData(testData);
    assertEquals(???,thd.maxTemperature(),0.01);
}
```

minHumidity() for BP and RTP

- Must test with concrete subclasses only
- Cannot instantiate abstract class directly
- Should produce the same answer since there is no difference between what Batch Processing vs Real-Time Processing compute
- Only difference is in *how* they compute it

```
@Test
public void testMinHumid1(){
    TempHumid thd = new TempHumidBP();
    thd.intakeData(testData);
    assertEquals(???,thd.minHumidity(),0.01);
}
@Test
public void testMinHumid2(){
    TempHumid thd = new TempHumidRTP();
    thd.intakeData(testData);
    assertEquals(???,thd.minHumidity(),0.01);
}
```

Test Timing Properties

- One of BP vs RTP should have faster intakeData()
- Other should have faster maxTemperature()/ minHumidity()
- Get current time (nanoTime()) both before and after operation
- Subtract to get deltaTime
- Compare (< or >?)
- Similarly test time for query methods

```
@Test
public void testTime1(){
    TempHumid thdBP = new TempHumidBP();
    TempHumid thdRTP = new TempHumidRTP();
    long time1 = System.nanoTime();
    thdBP.intakeData(testData);
    long time2 = System.nanoTime();
    long bpIntake = time2 - time1;

    long time3 = System.nanoTime();
    thdRTP.intakeData(testData);
    long time4 = System.nanoTime();
    long rtpIntake = time4 - time3;

    assertTrue(bpIntake ??? rtpIntake);
}
```

Make sure to check in with your lab leader if you are stuck!

ChatGPT

Here's a caricature showing the contrasting personalities of fast and slow temperature and humidity sensors. The fast sensor is depicted with a superhero vibe, showcasing its quick responsiveness, while the slow sensor appears more old-fashioned and sluggish, complete with snails to emphasize its slowness.

TempHumidRTPByDate

- Should ignore data not on the date we pass to the constructor
- How can you write a test that would expose a bug should the code accidentally consider all the data instead?

```
@Test
public void testMaxTemp3(){
    TempHumid thd = new TempHumidRTPByDate(20240401.0);
    thd.intakeData(testData);
    assertEquals(???,thd.maxTemperature(),0.01);
}

@Test
public void testMaxTemp4(){
    TempHumid thd = new TempHumidRTPByDate(20240402.0);
    thd.intakeData(testData);
    assertEquals(???,thd.maxTemperature(),0.01);
}
```