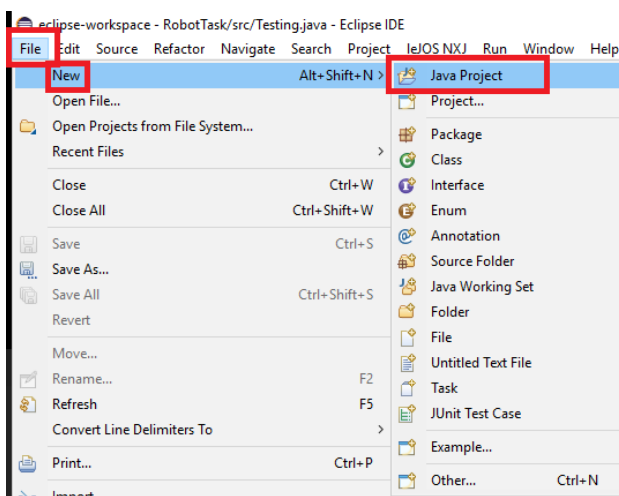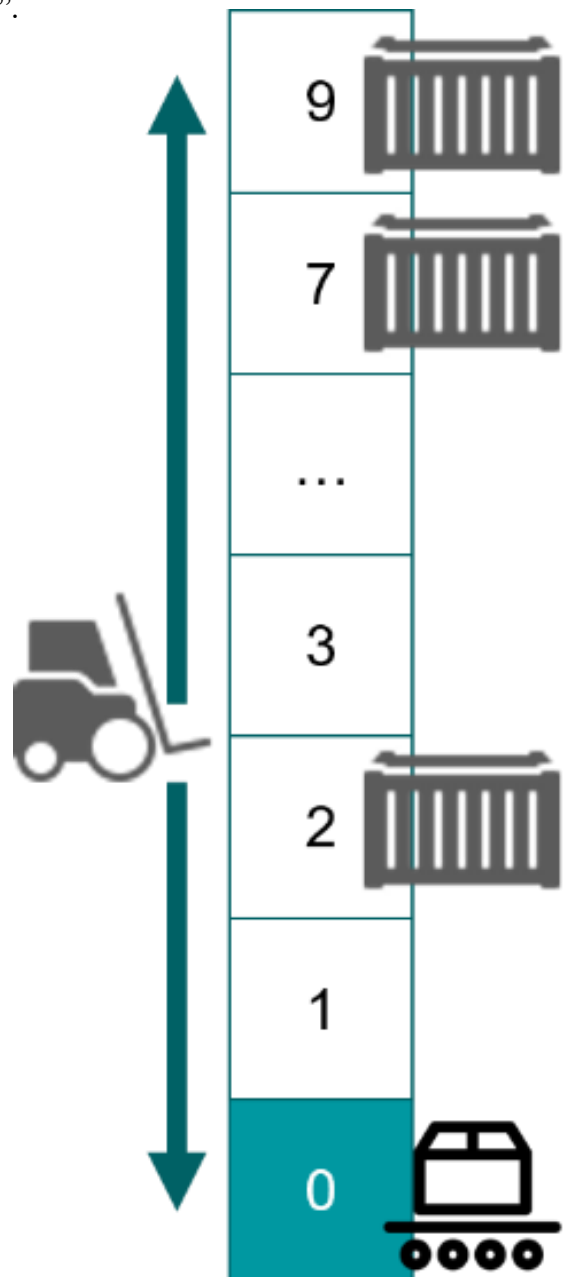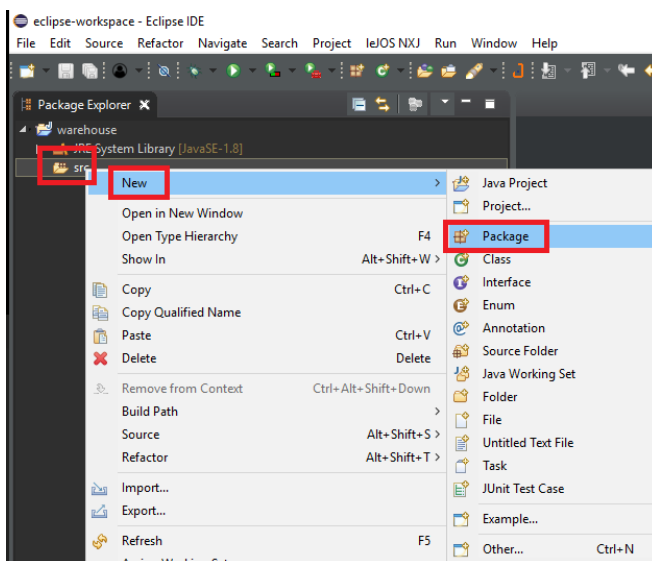# OPMS 2018 - Exercise 5
Automated high-bay warehouse

Task 0 (Setup)

- Create a new project in Eclipse and name it "Exercise5".

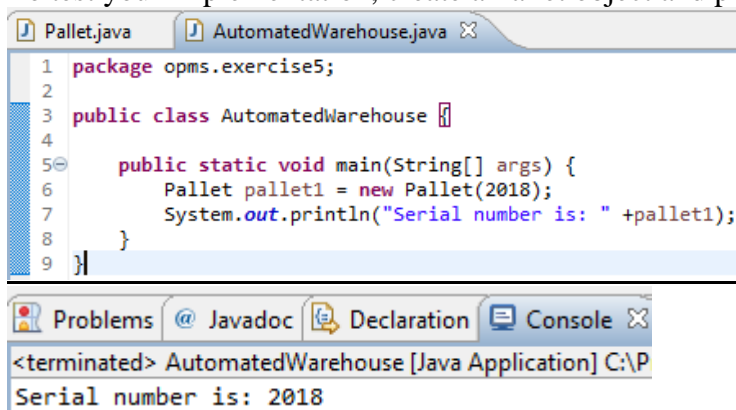- Create a package "opms.exercise5" and create all following classes inside this package.

Task 1 (Pallet)

1. Create a class `Pallet`. Add a field `private int serialNumber`.
2. Add a constructor `public Pallet(int serialNumber)` which saves the given serial number to the field `serialNumber`.
   Hint: Use `this.serialNumber = serialNumber` to set the Pallet's private field.
3. Add a method `public int getSerialNumber()` which returns the `serialNumber` field.
4. Add a method `public String toString()`, which returns the `serialNumber` as `String`.
   Hint:You can use `Integer.toString(serialNumber)` for that.
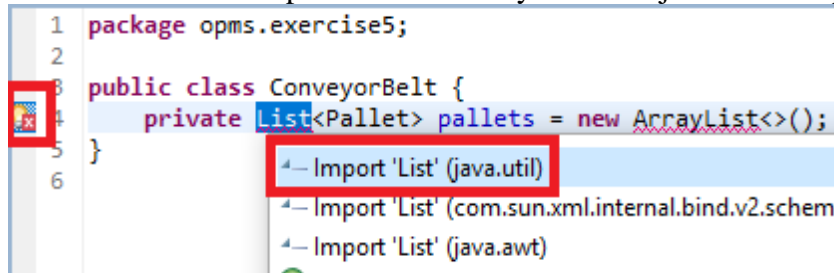5. Create a class AutomatedWarehouse and automatically generate a main-method (see screenshot).
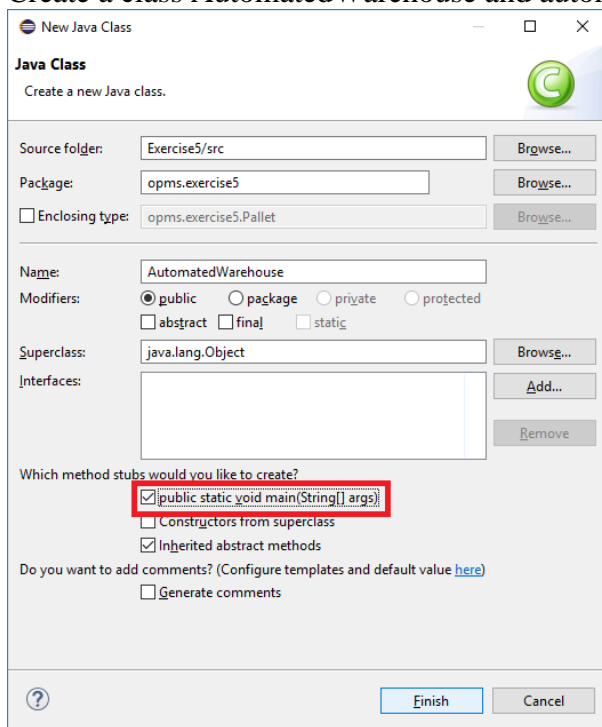
6. To test your implementation, create a Pallet object and print it to the console. Verify the output.

Task 2 (Conveyor Belt)

1. Create a class `ConveyorBelt`.
2. Add a field `private List<Pallet> pallets`.
   Initialize it via `new ArrayList<>()`.
   Hint: You need to import List and ArrayList from java.util. Eclipse helps you with that:
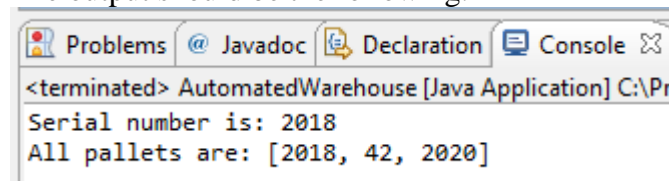
```
1  package opms.exercise5;
2
3  public class ConveyorBelt {
4      private List<Pallet> pallets = new ArrayList<>();
5  }
6
        ←— Import 'List' (java.util)
        ←— Import 'List' (com.sun.xml.internal.bind.v2.schem
        ←— Import 'List' (java.awt)
```

3. Add a method `public void loadPallet(Pallet pallet)` that adds the given `Pallet` to the List `pallets`.
   Hint: You can use a list's method `add(Pallet pallet)` for that.
4. Add a method `public List<Pallet> getPallets()` that returns the list called pallets.
5. To test your implementation, go to the main method in AutomatedWarehouse and add this:
   a. Create a ConveyorBelt instance and name it conveyorBelt
   b. Create two more Pallets pallet2 and pallet3, with the serialNumbers 42 and 2020
   c. Add all three Pallets to the conveyorBelt using the loadPallet method
   d. Print all pallets via the following command:
   ```
   System.out.println("All pallets are: "
   +conveyorBelt.getPallets());
   ```

   The output should be the following:

```
  Problems  @ Javadoc   Declaration   Console  ⊠
<terminated> AutomatedWarehouse [Java Application] C:\Pr
Serial number is: 2018
All pallets are: [2018, 42, 2020]
```

Task 3 (Forklift)

1. Create a class `Forklift`.
2. Add a field `private Pallet[] palletSlots`
   and a field `private ConveyorBelt conveyorBelt`.
3. Add a constructor `public Forklift(Pattet[] palletSlots, ConveyorBelt conveyorBelt)`, which saves both arguments to the private fields.
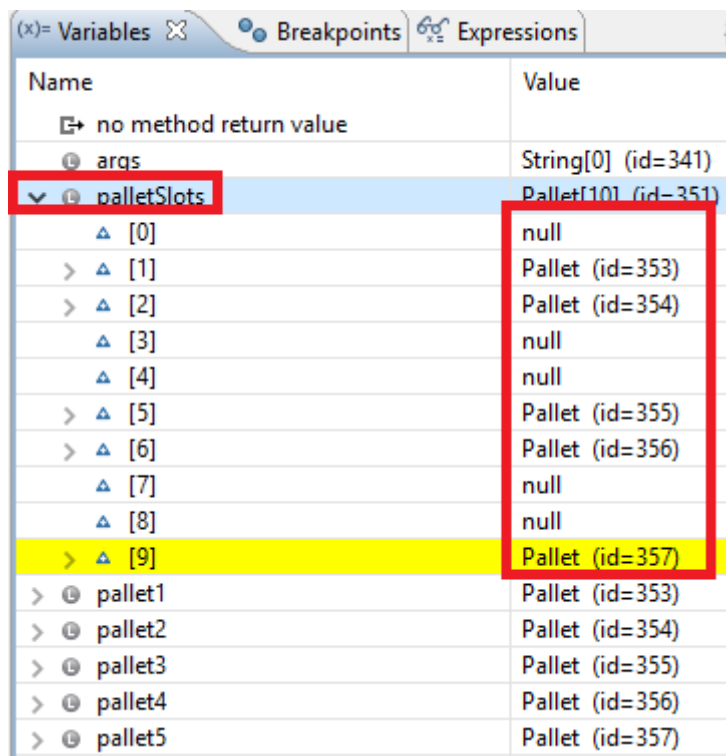   **Hint**: As in task 1, use the keyword `this` to assign the values.

Task 4 (Automated Warehouse)

1. Navigate to the main method in AutomatedWarehouse.
2. Comment-out all code in the main method you have written so far. Code marked as a comment will not be executed. Start the Comment with /* and end it with */

```
22    /*
23      This is a comment.
24    */
```

3. In your "empty" main method, create an Pallet[] array of size 10 and name it palletSlots.
   Hint: See slide 43 of module 4 for an example.
4. Create five Pallet objects via `new Pallet(int serialNumber)` with the serial numbers 541201, 541202, 663319, 663325 and 909042. Name them pallet1, pallet2, and so on.
5. Add these `Pallet` objects to the above created array at slots 1, 2, 5, 6 and 9. Do not assign anything to the other slots, so they will automatically contain null.
6. Also, create a `ConveyorBelt conveyorBelt = new ConveyorBelt();`
7. Finally, create a forklift instance via
   `Forklift forklift = new Forklift(palletSlots, conveyorBelt);`
8. Verify your implementation via the debugger. Add a breakpoint to where Pallet[] is created. Start the debugger and inspect your program line by line.
   When you went over `Pallet pallet5 = new Pallet(909042);` the debugger should look like this:
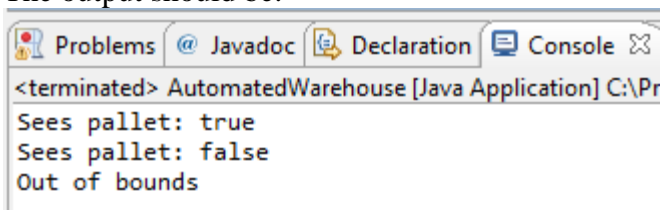
Task 5 (Bringing the Forklift to Life Part 1)

1. You will now enable the `Forklift` to move and shift `Pallets`.
2. Create a new Java class named `ForkliftOutOfBoundsException` that extends `Exception`.
3. Navigate to the `Forklift` class and add a field `private int position`.
4. Add a method `public void moveTo(int position) throws ForkliftOutOf-BoundsException`, which sets the private field `position` to the given value.
5. Before setting the position, check if the given position is in the `Forklift`'s boundaries (0-9). If not, throw a `new ForkliftOutOfBoundsException()`.
   Hint: Use `if` to check if the given position is smaller than 0 or greater than 9.
6. We want to know if there is a `Pallet` at the forklift's current position.
   Add a method `public boolean seesPallet()` that returns `true`, if there is a `Pallet` in the `palletSlots` at the forklift's current `position`, and `false` otherwise.
   Hint: If palletSlots has `null` at the position, then there is no pallet (return false).
7. Verify your implementation: Navigate to the main method in AutomatedWarehouse and add the following:

```
39          try {
40              forklift.moveTo(2);
41              System.out.println("Sees pallet: " +forklift.seesPallet());
42              forklift.moveTo(7);
43              System.out.println("Sees pallet: " +forklift.seesPallet());
44              forklift.moveTo(-11);
45          } catch (ForkliftOutOfBoundsException e) {
46              System.out.println("Out of bounds");
47          }
```

The output should be:

```
Problems  @ Javadoc  Declaration  Console
<terminated> AutomatedWarehouse [Java Application] C:\Pr
Sees pallet: true
Sees pallet: false
Out of bounds
```
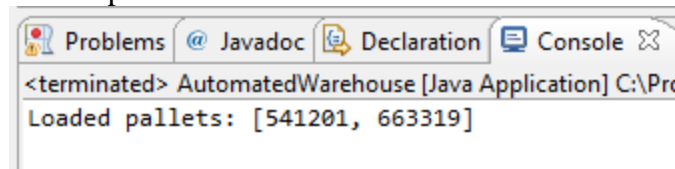
8. When done, delete the code you added in 7, it was only for verification.

Task 6 (Bringing the Forklift to Life Part 2)

1. We want to allow the forklift to lift pallets.
   Navigate to the forklift class and add a field Pallet `private Pallet currentPallet`.
2. Add a method `public void liftPallet()` to the forklift, that
   - Assigns the `Pallet` from the `palletSlots[position]` to the private field `currentPallet`
   - And removes the just picked up `Pallet` from the `palletSlots`.
     Hint: Assign `null` to the `palletSlots[position]` to remove the pallet.
3. We want to allow the forklift to place pallets on the conveyor belt.
4. Add a method
   `public void placeOnBelt() throws ForkliftOutOfBoundsException` that
   - Moves the forklift to the conveyor belt's position (0)
     Hint: You don't need to catch the exception thrown by moveTo() here.
   - Calls `conveyorBelt.loadPallet(this.currentPallet)` to load the current pallet on the belt.
   - Sets the `currentPallet` to `null` to stop carrying it.
5. Verify your implementation by adding code to your main method in AutomatedWarehouse:

```
51          try {
52              forklift.moveTo(1);
53              forklift.liftPallet();
54              forklift.placeOnBelt();
55
56              forklift.moveTo(5);
57              forklift.liftPallet();
58              forklift.placeOnBelt();
59
60              System.out.println("Loaded pallets: " +conveyorBelt.getPallets());
61          } catch (ForkliftOutOfBoundsException e) {
62              System.out.println("Out of bounds");
63          }
```

The output should look like this:

```
Problems  @ Javadoc  Declaration  Console ⋈
<terminated> AutomatedWarehouse [Java Application] C:\Pro
Loaded pallets: [541201, 663319]
```
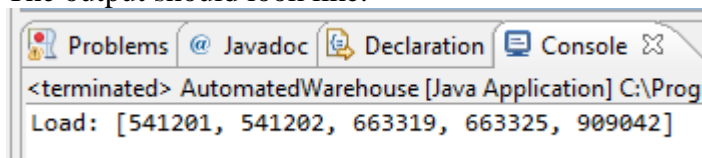
6. When done, delete the code you added in 5, it was only for verification.

Final Task 7 (Scheduling the Forklift)

1. In the final task, you need to load all pallets on the conveyor belt. Therefore, iterate over all pallet slots via `for(int slot=1; slot<=9; slot++)` and for each slot do:
   - Use `forklift.moteTo(slot)` to move the forklift to a pallet slot
   - Check if there is a pallet at its current position using the `seesPallet()` method.
   - If `true`: Call `liftPallet()` and then place it on the belt via `placeOnBelt()`.
   - If `false`: Do nothing.
2. Hint: You need to catch any `ForkliftOutOfBoundsExceptions` here, so wrap your for-loop into a `try-catch`-block.
3. After the for-loop, print a list of all loaded pallets to the console.
   Hint: Call `System.out.println("Load: "+conveyorBelt.getPallets())`.
   Hint: The `conveyorBelt.getPallets()` automatically calls `pallet.toString()` for each pallet for you.

   The output should look like:


   ```
   Problems  @ Javadoc  Declaration  Console
   <terminated> AutomatedWarehouse [Java Application] C:\Prog
   Load: [541201, 541202, 663319, 663325, 909042]
   ```

4. Verify your implementation as follows, re-run the program and inspect the output each time:
   - Changing the serial number of some pallets in the main method.
   - Add one or two additional pallets to the palletSlots array.
   - Remove some pallets from the palletSlots array.