Prerequisites

HTML

- The HTML body only has a toast container
- The reason for that is because each time a new notification is created and appended to this container, they will stack up after eachother. (This is bootstrap design)

```
<!-- BootStrap Toast Container -->
<div class="toast-container position-fixed top-0 end-0 p-3"></div>
```

Notification class

- The idea here is that we create an instance for each notification and define the type with a Factory Pattern, for staff and delivery, we have decided to do it this way to allow for scalability, in case there were to be implemented more classes in the future.
- **Container**: just the container DOM from HTML as shown above, we are appending the toast child to this container.
- **ID**: The toast div ID, so we can delete it later from the DOM.
- **Name**: Inheritance.
- **Surname**: Inheritance.
- **Message**: A custom message, such as how late they are.

```
class Notification {
  constructor(JSObject) {
    this.container = DOMUtils.getDOMElements.ui.toastContainer; //Just gets the
DOM container right away.
    this.id = JSObject.id; //We provide an ID for this div, which is in this
project Name.Surname of the instance.
    this.name = JSObject.name;
    this.surname = JSObject.surname;
    this.message = JSObject.message;
  }

  /**
   * Generates the default content for the notification.
   * Can be overridden by subclasses to provide specific content.
   */
  content() {
    return `
    <p>${this.name} ${this.surname}</p>
    <p>${this.message}</p>
    `;
```

```
  }

  /**
   * Displays the notification by:
   * 1. Creating a div element for the toast.
   * 2. Setting its content using the `content` method.
   * 3. Appending the div to the container.
   * 4. Using Bootstrap to show the toast.
   */
    Notify() {
    const div = DOMUtils.createDiv; //Creates a toast Div

    //Sets the attributes required for this div
    div.setAttribute('id', this.id);
    div.setAttribute('class', 'toast text-bg-danger');
    div.setAttribute('role', 'alert');
    div.setAttribute('aria-live', 'assertive');
    div.setAttribute('aria-atomic', 'true');
    div.setAttribute('data-bs-autohide', 'false');

    //Additional inner divs, including header, close btn and body
    div.innerHTML = `
        <div class="toast-header text-bg-danger border-0">
          <button type="button" class="btn-close btn-close-white me-2 m-auto"
data-bs-dismiss="toast" aria-label="Close"></button>
        </div>
        <div class="toast-body">
          ${this.content()}
        </div>
    `;

    //Appends the div to the container
    this.container.appendChild(div);

    //Initializes bootstrap to create a notification and show it
    DOMUtils.createToast(this.id);
  }
}

export class StaffNotification extends Notification {
  constructor(JSObject) {
    super(JSObject);
    this.picture = JSObject.picture;
  }

  /**
   * Polymorphism
   * Overrides the base `content` method to include a picture
   * and a specific late message for staff notifications.
   */
  content() {
    return `
    <img src="${this.picture}" alt="Staff Picture">
```

```
      <p>${this.name} ${this.surname} is late!</p>
      <p>${this.message}</p>
      `;
    }
  }

  export class DeliveryNotification extends Notification {
    constructor(JSObject) {
      super(JSObject);
      this.phone = JSObject.phone;
      this.adress = JSObject.adress;
      this.return = JSObject.return;
    }

    /**
     * Polymorphism
     * Overrides the base `content` method to include delivery-specific details.
     */
    content() {
      return `
      <p>${this.name} ${this.surname}</p>
      <p>Return time was: ${this.return}</p>
      <p>Address: ${this.adress}</p>
      <p>Phone: ${this.phone}</p>
      <p>${this.message}</p>
    `;
    }
  }
```

Helper function

- Responsible for showing the toast and removing it from the DOM after is dismissed.

```
    createToast(id) {
      const toastWindow = document.getElementById(`${id}`);
      const toastBootstrap = bootstrap.Toast.getOrCreateInstance(toastWindow);
      toastBootstrap.show();

      toastWindow.addEventListener('hidden.bs.toast', () => {
        toastWindow.remove(); //Removes the created DOM element once the toast
has faded or closed manually by the user
      });
    }
```

Usage

Creating a toast object and calling it from the a staff instance:

```
staffMemberIsLate(EMPLOYEES) {
    const { lateInterval } = EMPLOYEES.get('config');

    const checkIfLate = setInterval(() => {
      const time = factory.createEmployee('time', new Date());
      const late = time.isLate(this.expectedRTime);

      if (this.status === 'Out') {
        if (late) {

          //Create toast notification data and message
          const toastData = {
            id: this.id,
            picture: this.picture,
            name: this.name,
            surname: this.surname,
            message: `Late by: ${time.lateBy(this.expectedRTime)} mins`
          };

          const toastInstance = factory.createEmployee('staffNotification',
toastData);
          toastInstance.Notify();

          clearInterval(checkIfLate);
        }
      } else {
        clearInterval(checkIfLate);
      }
    }, lateInterval); //Change the interval in WDT_APP.js Settings
    return checkIfLate;
  }
```

Creating a toast object and calling it from the a delivery instance:

```
deliveryDriverIsLate(EMPLOYEES) {
    const { lateInterval } = EMPLOYEES.get('config');
    const deliveries = EMPLOYEES.get('deliveries');

    const checkIfLate = setInterval(() => {
      const time = factory.createEmployee('time', new Date());
      const late = time.isLate(this.expectedRTime);
      const deliveryID = this.id;

      if (deliveryID in deliveries) {
        if (late) {

          //Create toast notification data and message
          const toastData = {
            id: deliveryID,
            name: this.name,
```

```
            surname: this.surname,
            phone: this.phone,
            adress: this.adress,
            return: this.expectedRTime,
            message: `Late by: ${time.lateBy(this.expectedRTime)} mins`
          };

          const toastInstance = factory.createEmployee('deliveryNotification',
toastData);
          toastInstance.Notify();
          clearInterval(checkIfLate);
        }
      } else {
        clearInterval(checkIfLate);
      }
    }, lateInterval); //Change the interval in WDT_APP.js Settings
    return checkIfLate;
  }
```