## Oppgave 2: kommentering

«Lag gode kommentarer til koden du har utviklet i prosjektet ditt.»

Til prosjektet skriver vi både koden og kommentarene på engelsk. Vi sikter mot å ha beskrivende kommentarer i toppen av funksjonene og klassene vi lager, samt kommentarer for kode «blokker» som gjør mer komplekse operasjoner. Her er et eksempel på hvordan vi tenker å bruke kommentarer i prosjektet:

```php
/**
 * This static method returns a PHP Data Object (PDO) connection to the database.
 * Note, remember to add DBConfig.php with connection constants.
 *
 * @return PDO|null Returns a PDO connection to the database, or null if an exception is thrown.
 * @throws PDOException Throws a PDOException if the connection fails.
 *
 */
6 usages  Carsten Østergaard
public static function getConnection(): ?PDO
{
    try {
        $conn = new PDO(
            dsn: "mysql:host=" . DBConfig::DB_HOST_NAME . ";dbname=" . DBConfig::DB_NAME,
            username: DBConfig::DB_NAME,
            password: DBConfig::DB_PASSWORD);

        $conn->setAttribute( attribute: PDO::ATTR_ERRMODE,  value: PDO::ERRMODE_EXCEPTION);
        $conn->exec( statement: "set names utf8");
    } catch (PDOException $exception) {
        // TODO add logging
        // TODO do not echo exceptions to the user
        echo "Connection error: " . $exception->getMessage();
    }

    return $conn ?? null;
}
```

Kommentarene over funksjonen inneholder en kort beskrivelse av hva den gjør, og returnerer. Dette fungerer som en slags kort dokumentasjon på funksjonen.

Kommentaren under nevner at det er for modul 6 oppgave 5, denne har gruppen fått tilgang på fra andre studenter. Gruppen mener dog at funksjonen ‘isValid’ har for mye ansvar, og burde brytes ned i tre ulike funksjoner for å forminske ‘coupling’ og høyne ‘cohesion’.

```php
/**
 * This function is made for Module 6, task 5 to validate
 * either an email, password or a phone number in the same function.
 *
 * @param string $type must be string of one of these: email | password | phone.
 * @param string $value The string to check if is valid.
 * @return bool True if valid, false if not.
 * @throws Exception Will throw an exception if type entered is wrong.
 */
1 usage  Carsten Østergaard
public static function isValid(string $type, string $value): bool
{
    return match ($type) {
        'email' => filter_var($value, filter: FILTER_VALIDATE_EMAIL) !== false,
        'password' => self::validatePassword($value),
        'phone' => preg_match( pattern: "/^[0-9]{8}$/", $value),
        default => throw new Exception( message: 'Invalid validation type specified'),
    };
}
```

```php
/**
 * Checks if the password is valid.
 *
 * @param string $password The password to check.
 * @return bool true if the password is valid, false if the password is invalid.
 * TODO make this return an array to give a message of what failed to the user
 */
1 usage  Carsten Østergaard *
private static function validatePassword(string $password): bool {
    return strlen($password) >= 9 &&                    // Longer than, or 9 characters.
        preg_match( pattern: '/[0-9]/', $password) &&     // Has one or more numbers.
        preg_match( pattern: '/[A-Z]/', $password) &&     // Has one or more upper case letters.
        preg_match( pattern: '/[a-z]/', $password) &&     // Has one or more lower case letters.
        preg_match( pattern: '/[^a-zA-Z0-9]/', $password);// Has one or more special characters.
}
```