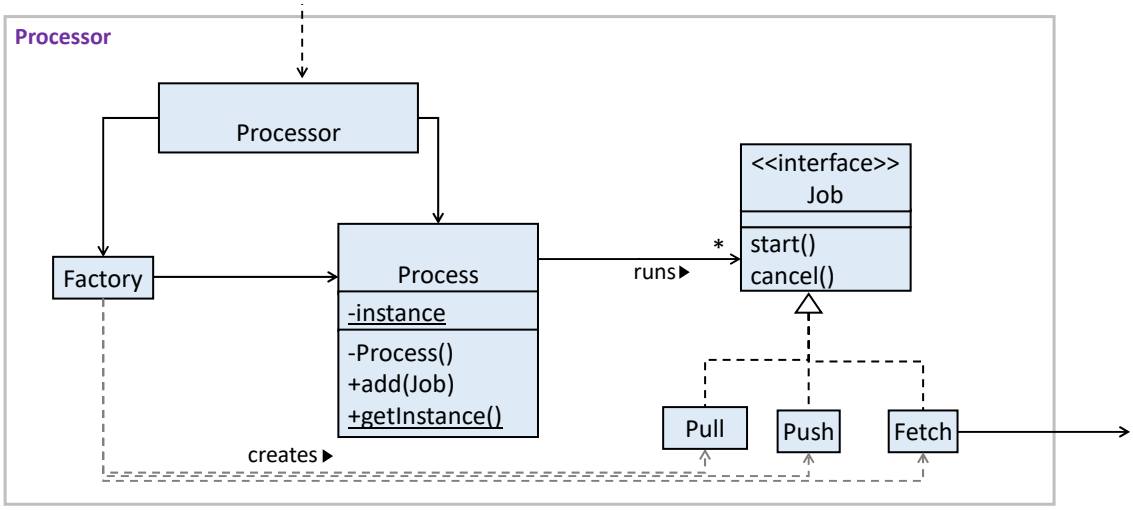
Q1. Which design patterns are used in the following design?

[ ] Façade [ ] Singleton [ ] Command



Q2. Suggest at least 5 ways (no more than 2 related to the coding standard) to improve the quality of the following code.

```
L1  /* Set user as 'active' on the server.
L2   * @throws CannotActivateException if the user doesn't exist on the server
L3   */
L4  void activateUserOnServer(String userName) throws CannotActivateException{
L5      log("trying to activate");
L6      assert isUsernameAcceptable(userName);
L7      ServerConnection.activate(userName);
L8      if(!ServerConnection.isActivated(userName))
L9         throw new CannotActivateException(userName + " not activated");
L10     Account account = AccountManager.getAccount(userName);
L11     account.toggleActivatedStatus(); //mark as activated
L12 }
```

Q3 (a) Design *unit* test cases for unit testing the `Parser#isValidName` method below.

```
/* Returns true if the name is non-empty and
   not null and not longer than 40 chars. */
public static boolean isValidName(String name) {
    // ...
}
```

Input	Expected output
null	false

(b) Design *integration* test cases for the method below. Note that it uses the method given in (a).

```
/* Throws StorageException if name is not valid
   or if name already exists in the database. */
public void saveScore(String name) throws StorageException {
    if (!Parser.isValidName(name)) {
      throw new StorageException("invalid name");
    }
    if (storage.isFound(name)) {
       throw new StorageException("already exists");
    }
    storage.save(name);
}
```

Input	Expected output