Spring Au 2021 - February(Batch -2)
Design Patterns - Morning

Name - Nitin Kaushik
Github - https://github.com/nitinkaushik10/SAU-Feb-Batch-2/

Ques1:

Creational
1. Singleton
2. Builder
3. Static factory method
4. Abstract factory

Structural
1.Flyweight
2.Adapter
3.Decorator

Behavioural
1. Chain of responsibility
2. Command
3. Iterator
4. Strategy
5. Template method
6. Observer

Ques2:

a) Similarly, the object of the AccessChecker class are instantiated in the ServerConfig Class which in turn means that it is tightly coupled to the ServerConfig Class. The ServerConfig class has methods like setting and loading the configuration file, etc, so it might not be practical to create instances of them for the tests. AS the ServerConfig follows the Singleton design, we can override its getInstance method. So, instead of going with the classes, we should go with declaring the interfaces.

b) The interface for ServerConfig will be :

```
public interface ServerConfigInterface
{
    public String getAccessLevel(User u);
}
```

The interface for AccessCheckerInterface will be :

```
public interface AccessCheckerInterface
{
    public boolean mayAccess(User user, String path);
}
```

Ques3:

```
public class MainTest {

 public static void main(String[] args) {
```

```
            Module module = new
              AbstractModule() {
              @Override
              protected void configure() {
                bind(AccessCheckerInterface.class).to(Access
                CheckerMock.class);

              }

            };

            SessionManager mgr =
            Guice.createInjector(module).getInstance(SessionManager.class);
            User user = new User();

            mgr.createSession(user, "any path");

        }

}
```

We can create the static factory class in

the following way - public class

Responses {

```
    public static Response notFoundResponse() {

        return new NotFoundResponse();

    }

    public static Response markdownResponse() {
```

```
        return new MarkdownResponse();
}
    public static Response fileResponse() {

        return new FileResponse();
} }

    public class
        Response
        { private
        String
        status;
    private Map<String,
    String> headers; private
    String body;

}
public class Responses {

    public static Response response(String status, Map<String, String>
    headers, String body) {
```

```
        return new Response(status, headers, body);
    }

        public static Response file(String
          status, String path) { Path filePath
          = Paths.get(path);
        HashMap<String, String> headers = new
        HashMap<String, String>();
        headers.put("content-type",
        Files.probeContentType(filePath));
        byte[] bytes =
        Files.readAllBytes(filePath);
        String body = new
        String(bytes);

        return response(status, headers, body);
    }
    public static Response notFound() {

        return file("404", app.Assets.getInstance().getNotFoundPage());

    }

    public static markdown(String body) {
```

```
    HashMap<String, String> headers = new
    HashMap<String, String>();
    headers.put("content-type", “text/html");

    return response("200", headers, Markdown.parse(body).toHtml());

}
}
```