LaTeX

Написать исходный код CDI бина, реализующего паттерн «команда» (Command Pattern)

```
interface Command {void execute();}
@Named(value="cmd1") @ApplicationScoped
public class Cmd1 implements Command {
    void execute() { ... };}
@Named(value="cmd2") @ApplicationScoped
public class Cmd2 implements Command {
    void execute() { ... };}
@Named @ApplicationScoped
public class MyBean implements Command {
    private final Command cmd1, cmd2;
    @Inject
    public MyBean(@Named("cmd1") Command cmd1,
        ↪ @Named("cmd2") Command cmd2) {
        ↪ this.cmd1 = cmd1; this.cmd2 = cmd2;}
    public void cmd(int n) {if (n == 1)
        ↪ cmd1.execute(); if (n == 2)
        ↪ cmd2.execute();}}
```

JSF страница, динамически подгружаемая и выводящая новостную ленту с новостями формата: автор, заголовок, дата, иллюстрация, аннотация и полный текст(показывается при нажатии на соответствующую строчку)

```
<h:body> <h:dataTable value="#{newsBean.newsList}"
      ↪ var="news" border="1">
<h:column> <f:facet name="header">Author</f:facet>
<h:outputText value="#{news.author}" /> </h:column>
<h:column> <f:facet name="header">Title</f:facet>
<h:outputText value="#{news.title}" /> </h:column>
<h:column> <f:facet name="header">Date</f:facet>
<h:outputText value="#{news.date}" /> </h:column>
<h:column> <f:facet name="header">Image</f:facet>
<h:graphicImage value="#{news.image}" width="100"
      ↪ height="100" alt="News Image" />
</h:column> <h:column> <f:facet
      ↪ name="header">Summary</f:facet>
<h:outputText value="#{news.summary}" /> </h:column>
<h:column> <f:facet name="header">Full
      ↪ Text</f:facet>
<h:commandLink value="Show Full Text"
      ↪ action="#{newsBean.showFullText(news)}">
<f:setPropertyActionListener
      ↪ target="#{newsBean.selectedNews}"
      ↪ value="#{news}" />
</h:commandLink> </h:column> </h:dataTable>
<h:panelGroup rendered="#{not empty
      ↪ newsBean.selectedNews}">
<h3>Full Text of News:</h3>
<h:outputText
      ↪ value="#{newsBean.selectedNews.fullText}"
      ↪ escape="false" />
</h:panelGroup> </h:body>
```

Написать React компонент формирующий таблицу пользователей, данные приходят в props

```
function UsersTable(props) { return (
<table className="table"> <thead>
<tr><td>Name</td><td>Surname</td></tr>
</thead> <tbody> { props.data.map((user, i) => (
<tr key={i}> <td>{user.name}</td>
<td>{user.surname}</td> </tr>))} </tbody> </table>)}
```

Написать веб-приложение на JSF (xhtml + CDI-бин) со списком студентов и бин, который будет реализовывать логику отчисления студентов. Напротив каждого имени студента должна быть кнопка "отчислить". Обновление должно производиться при помощи AJAX

```
@Named
@ApplicationScoped
public class StudentBean implements Serializable {
private List<String> students;
public List<String> getStudents(){ return students;}
public void expelStudent(String studentName) {
    students.remove(studentName);}}

<h:body><h:form>
<ui:repeat value="#{studentBean.students}"
      ↪ var="student"> #{student}
<h:commandButton value="Expel" action="#{stude
      ↪ ntBean.expelStudent(student)}">
<f:ajax execute="@this" render="@form" />
</h:commandButton>
<br/></ui:repeat></h:form></h:body>
```

Интерфейс на React, формирующий две страницы с разными URL: Главную (/home) и Новости (/news). Переход между страницами должен осуществляться посредством гиперссылок.

```
export function App(props) {
return (<BrowserRouter>
<Routes><Route path="/home"
      ↪ element={<div><h1>Home</h1> <a
      ↪ href='/news' > news</a></div>} />
<Route path="/news" element={<div><h1>news</h1> <a
      ↪ href='/home' > home</a></div>} />
</Routes></BrowserRouter>);}
```

Написать CDI Bean калькулятор, поддерживающий 4 базовые операции для целых чисел

```
@Named(value="calc")
@ApplicationScoped
public class Calc implements Serializeable{
public int add(int a, int b) { return a + b; }
public int sub(int a, int b) { return a - b; }
public int mul(int a, int b) { return a * b; }
public int div(int a, int b) { return a / b; }}
```

Интерфейс JSF (xhtml страница + CDI), реализующий ввод паспортных данных (серия, номер, дата, место выдачи)

```
@ManagedBean @SessionScoped @Setter @Getter
public class PassportInputBean {
private String series; private String number;
private Date date; private String place;
@ManagedProperty(value = "#{repoBean}")
private PassportRepo repo;
public void storePassportData() {
repo.store(series, number, date, place);}}
<h:form>Enter series
<h:inputText type="text"
      ↪ value="#{passportInputBean.series}"
required="true"/>Enter number
<h:inputText type="text"
      ↪ value="#{passportInputBean.number}">
Enter date<h:inputText
      ↪ value="#{passportInputBean.date}">
<f:convertDateTime pattern="yyyy-MM-dd"/>
</h:inputText> Enter born place
<h:inputText type="text"
      ↪ value="#{passportInputBean.place}">
<h:commandButton value="Send"
      ↪ action="#{passportInputBean.storePas
sportData()}"> </h:form>
```

Привести фрагмент кода управляемого бина, увеличивающего на 1 значение, отображаемое на кнопке при каждом клике по ней

```
@ManagedBean @ApplicationScoped
public class MyBean implements Serializable {
private int value = 0; public void increment() {
value++;} public int getValue() {return value;}
public void setValue(int value) {this.value =
      ↪ value;}}
<h:commandButton action= "#{myBean.increment}"
      ↪ value = "#{bean.value}"/>
```

Сделать бин который показывает время в минутах со старта сервера

```
@Named("serverTimer") @ApplicaitonScoped
public class ServerTimer { private long start;
public ServerTimer() {start =
      ↪ System.currentTimeMillis();}
public long getTime() {return
      ↪ (System.currentTimeMillis() - start) /
      ↪ 60000;}}
```

Написать страницу JSF, которая бы выводила сначала 10 простых чисел, а затем с помощью Ajax запроса ещё 10.

```
@ApplicationScoped @Named public class
      ↪ PrimeNumber{@Getter
private final List<Long> primes = new ArrayList<>();
public PrimeNumber(){nextPrimes();}
public nextPrimes(){//add in primes next 10 prime
      ↪ numbers}}

<h:dataTable id="table"
      ↪ value"#{primeNumber.primers}" var="prime">
<h:column> <h:outputText value="#{prime}"/>
</h:column> </h:dataTable> <h:form>
<h:commandButton value="Next 10"
      ↪ action="#{primeNumber.nextPrimes}">
<f:ajax execute="@form" render="table"/>
</h:commandButton> </h:form>
```

JSF Manager Bean, после инициализации HTTP-сессии формирующий коллекцию с содержимым таблицы H_УЧЕБНЫЕ_ПЛАНЫ. Для доступа к БД необходимо использовать JDBC-ресурс jdbc/OrbisPool.

```
@Named("myBean") @SessionScoped class MyBean {
private List<String> plans;
private List<String> getPlans() {return plans;}
@Resource(name="jdbc/OrbisPool",type=DataSource.class)
private DataSource dataSource; @PostConstruct
private void loadPlans() {
try (var conn = dataSource.getConnection()) {
var stmt = conn.createStatement();
var rs = stmt.executeQuery("SELECT name FROM
      ↪ plans");
plans = new ArrayList<>(); while (rs.next()) {
plans.add(rs.getString("name"));}}}}
```

Написать managed bean и задать ему scope такой же как у бина otherBean

```
@ManagedBean @ApplicationScoped
public class OtherBean {
    @ManagedProperty(value="#{myBean}")
@Getter private MyBean myBean;}
@ManagedBean @NoneScoped @Named
public class MyBean{}
```

Vue.js простейший чат бот, который на любое сообщение отвечает сам дурак

```
<template> <input v-model="message_value"
      ↪ placeholder="message"/>
<button @click="send">send</button>
<li v-for="item in message_history" :key="item">
{{item}}</li></template>
<script> export default { data() { return {
message_value: '', message_history: []}},methods: {
send(){this.message_history.push(this.message_value);
this.message_history.push("sam durak");
this.message_value = "";}}}</script>
```

RestController, который реализует перевод градусов Цельсия в Фаренгейты и обратно

```
@RestController class TempController {
@GetMapping("/convert/c/f")
double cToF(@RequestParam double c) {return c * 1.8
      ↪ + 32.0;}
@GetMapping("/convert/f/c")
double fToC(@RequestParam double f) {return (f -
      ↪ 32.0) / 1.8;}}
```

Написать JSF страницу с многострочным полем, в которое можно вводить только строчные символы латиницы.

```
<h:head> <script> function validateInput() {
var textArea =
      ↪ document.getElementById('inputTextArea');
var regex = /^[a-z\s]*$/; if
      ↪ (!regex.test(textArea.value)) {
textArea.value = '';}}</script></h:head>
<h:body><h:form>
<h:inputTextarea id="inputTextArea" rows="5"
      ↪ cols="30" oninput="validateInput()"
      ↪ /></h:form></h:body>
```