МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)

УТВЕРЖДАЮ
Директор департамента образования

Документ подписан электронной подписью
Сертификат: 1c6cfa0a-52a6-4f49-aef0-5584d3fd4820
Владелец: Троян Павел Ефимович
Действителен: с 19.01.2016 по 16.09.2019

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Конструирование программного обеспечения

Уровень образования: **высшее образование - бакалавриат**
Направление подготовки / специальность: **09.03.04 Программная инженерия**
Направленность (профиль) / специализация: **Проектирование и разработка программных продуктов**
Форма обучения: **заочная**
Факультет: **ЗиВФ, Заочный и вечерний факультет**
Кафедра: **АОИ, Кафедра автоматизации обработки информации**
Курс: **3, 4**
Семестр: **6, 7**
Учебный план набора 2014 года

Распределение рабочего времени

№	Виды учебной деятельности	6 семестр	7 семестр	Всего	Единицы
1	Лекции	2	4	6	часов
2	Лабораторные работы	0	8	8	часов
3	Всего аудиторных занятий	2	12	14	часов
4	Самостоятельная работа	34	20	54	часов
5	Всего (без экзамена)	36	32	68	часов
6	Подготовка и сдача зачета	0	4	4	часов
7	Общая трудоемкость	36	36	72	часов
				2.0	З.Е.

Контрольные работы: 7 семестр - 1
Зачет: 7 семестр

Томск 2018

ЛИСТ СОГЛАСОВАНИЯ

Рабочая программа дисциплины составлена с учетом требований федерального государственного образовательного стандарта высшего образования (ФГОС ВО) по направлению подготовки (специальности) 09.03.04 Программная инженерия, утвержденного 12.03.2015 года, рассмотрена и одобрена на заседании кафедры АОИ «___» _____ 20__ года, протокол №_____.

Разработчик:

старший преподаватель каф. АОИ _____ И. В. Безходарнов

Заведующий обеспечивающей каф. АОИ _____ Ю. П. Ехлаков

Рабочая программа дисциплины согласована с факультетом и выпускающей кафедрой:

Декан ЗиВФ _____ И. В. Осипов

Заведующий выпускающей каф. АОИ _____ Ю. П. Ехлаков

Эксперты:

Доцент кафедры автоматизации обработки информации (АОИ) _____ Н. Ю. Салмина

Доцент кафедры автоматизации обработки информации (АОИ) _____ А. А. Сидоров

1. Цели и задачи дисциплины

1.1. Цели дисциплины

Научиться разбираться в различных подходах к конструированию программного обеспечения.

Овладеть навыками конструирования различных типов программного обеспечения.

1.2. Задачи дисциплины

– Изучение принципов проектирования программного обеспечения.
– Изучение шаблонов проектирования программного обеспечения.
– Изучение приемов и методов решения типовых задач, возникающих при конструировании программного обеспечения.

2. Место дисциплины в структуре ОПОП

Дисциплина «Конструирование программного обеспечения» (Б1.В.ОД.18) относится к блоку 1 (вариативная часть).

Предшествующими дисциплинами, формирующими начальные знания, являются: Объектно-ориентированный анализ и программирование.

Последующими дисциплинами являются: Технологии программирования.

3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций:

– ПК-3 владением навыками использования различных технологий разработки программного обеспечения;

В результате изучения дисциплины обучающийся должен:

– **знать** Различные подходы (методики) конструирования программного обеспечения.
– **уметь** Применять различные методики конструирования программного обеспечения на практике.
– **владеть** Навыками проектирования и создания архитектуры программного обеспечения.

4. Объем дисциплины и виды учебной работы

Общая трудоемкость дисциплины составляет 2.0 зачетных единицы и представлена в таблице 4.1.

Таблица 4.1 – Трудоемкость дисциплины

Виды учебной деятельности	Всего часов	Семестры	
		6 семестр	7 семестр
Аудиторные занятия (всего)	14	2	12
Лекции	6	2	4
Лабораторные работы	8	0	8
Самостоятельная работа (всего)	54	34	20
Оформление отчетов по лабораторным работам	4	0	4
Подготовка к лабораторным работам	4	0	4
Проработка лекционного материала	2	1	1
Самостоятельное изучение тем (вопросов) теоретической части курса	33	33	0
Выполнение контрольных работ	11	0	11
Всего (без экзамена)	68	36	32
Подготовка и сдача зачета	4	0	4
Общая трудоемкость, ч	72	36	36

Зачетные Единицы	2.0		

5. Содержание дисциплины

5.1. Разделы дисциплины и виды занятий

Разделы дисциплины и виды занятий приведены в таблице 5.1.

Таблица 5.1 – Разделы дисциплины и виды занятий

Названия разделов дисциплины	Лек., ч	Лаб. раб., ч	Сам. раб., ч	Всего часов (без экзамена)	Формируемые компетенции
6 семестр					
1 Методики конструирования программного обеспечения	2	0	34	36	ПК-3
Итого за семестр	2	0	34	36	
7 семестр					
2 Методики конструирования программного обеспечения	4	8	20	32	ПК-3
Итого за семестр	4	8	20	32	
Итого	6	8	54	68	

5.2. Содержание разделов дисциплины (по лекциям)

Содержание разделов дисциплин (по лекциям) приведено в таблице 5.2.

Таблица 5.2 – Содержание разделов дисциплин (по лекциям)

Названия разделов	Содержание разделов дисциплины (по лекциям)	Трудоемкость, ч	Формируемые компетенции
6 семестр			
1 Методики конструирования программного обеспечения	Обзорная лекция: методики конструирования программного обеспечения. Постановка задачи на самостоятельное изучение методик.	2	ПК-3
	Итого	2	
Итого за семестр		2	
7 семестр			
2 Методики конструирования программного обеспечения	Конструирование бизнес-логики программного обеспечения: методики, примеры, области применения.Проектирование интерфейсов программного обеспечения: методики, примеры.Планирование среды эксплуатации программного обеспечения: примеры организации.	4	ПК-3
	Итого	4	
Итого за семестр		4	
Итого		6	

5.3. Разделы дисциплины и междисциплинарные связи с обеспечивающими (предыдущими) и обеспечиваемыми (последующими) дисциплинами

Разделы дисциплины и междисциплинарные связи с обеспечивающими (предыдущими) и обеспечиваемыми (последующими) дисциплинами представлены в таблице 5.3.

Таблица 5.3 – Разделы дисциплины и междисциплинарные связи

Наименование дисциплин	№ разделов данной дисциплины, для которых необходимо изучение обеспечивающих и обеспечиваемых дисциплин	
	1	2
Предшествующие дисциплины		
1 Объектно-ориентированный анализ и программирование	+	+
Последующие дисциплины		
1 Технологии программирования	+	+

5.4. Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий

Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий представлено в таблице 5.4.

Таблица 5.4 – Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий

Компетенции	Виды занятий			Формы контроля
	Лек.	Лаб. раб.	Сам. раб.	
ПК-3	+	+	+	Контрольная работа, Отчет по лабораторной работе, Зачет, Тест

6. Интерактивные методы и формы организации обучения

Не предусмотрено РУП.

7. Лабораторные работы

Наименование лабораторных работ приведено в таблице 7.1.

Таблица 7.1 – Наименование лабораторных работ

Названия разделов	Наименование лабораторных работ	Трудоемкость, ч	Формируемые компетенции
7 семестр			
2 Методики конструирования программного обеспечения	Конструирование бизнес-логики и архитектуры простого приложения с применением объектно-ориентированного проектирования.	4	ПК-3
	Применение шаблонов проектирования программного обеспечения.	4	
	Итого	8	
Итого за семестр		8	

Итого	8	

8. Практические занятия (семинары)

Не предусмотрено РУП.

9. Самостоятельная работа

Виды самостоятельной работы, трудоемкость и формируемые компетенции представлены в таблице 9.1.

Таблица 9.1 – Виды самостоятельной работы, трудоемкость и формируемые компетенции

Названия разделов	Виды самостоятельной работы	Трудоемкость, ч	Формируемые компетенции	Формы контроля
6 семестр				
1 Методики конструирования программного обеспечения	Самостоятельное изучение тем (вопросов) теоретической части курса	33	ПК-3	Тест
	Проработка лекционного материала	1		
	Итого	34		
Итого за семестр		34		
7 семестр				
2 Методики конструирования программного обеспечения	Выполнение контрольных работ	11	ПК-3	Контрольная работа, Отчет по лабораторной работе, Тест
	Проработка лекционного материала	1		
	Подготовка к лабораторным работам	4		
	Оформление отчетов по лабораторным работам	4		
	Итого	20		
Итого за семестр		20		
	Подготовка и сдача зачета	4		Зачет
Итого		58		

10. Курсовой проект / курсовая работа

Не предусмотрено РУП.

11. Рейтинговая система для оценки успеваемости обучающихся

Рейтинговая система не используется.

12. Учебно-методическое и информационное обеспечение дисциплины

12.1. Основная литература

1. Введение в архитектуру программного обеспечения [Электронный ресурс]: Учебное пособие / Гагарина Л.Г., Федоров А.Р., Федоров П.А. - М. ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 320 с. 60x90 1/16. - (Высшее образование) (Переплёт 7БЦ) ISBN 978-5-8199-0649-1 - Режим доступа: http://znanium.com/catalog/product/542665 (дата обращения: 06.08.2018).

2. Технология разработки программного обеспечения [Электронный ресурс]: Учеб. пос. / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул; Под ред. проф. Л.Г.Гагариной - М. ИД ФОРУМ НИЦ

Инфра-М, 2013. - 400 с. ил.; 60x90 1/16. - (Высшее обр.). (п) ISBN 978-5-8199-0342-1 - Режим доступа: http://znanium.com/catalog/product/389963 (дата обращения: 06.08.2018).

12.2. Дополнительная литература

1. Орлов С.А., Цилькер Б.Я., Технологии разработки программного обеспечения: современный курс по программной инженерии: Учебник для вузов. 4-е изд. – СПб.: Питер, 2012. – 608 с. ГРИФ (наличие в библиотеке ТУСУР - 15 экз.)
2. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления. Учебное пособие для вузов. – М.: Горячая линия - Телеком, 2011 – 304 с. ГРИФ (наличие в библиотеке ТУСУР - 20 экз.)
3. Грекул В. И. Проектирование информационных систем. Курс лекций : Учебное пособие для вузов / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. - М. : Интернет-Университет Информационных Технологий, 2005. - 298[5] с. (наличие в библиотеке ТУСУР - 20 экз.)

12.3. Учебно-методические пособия

12.3.1. Обязательные учебно-методические пособия

1. Конструирование программного обеспечения [Электронный ресурс]: Методические указания к лабораторным работам и организации самостоятельной работы / И. В. Безходарнов - 2018. 14 с. - Режим доступа: https://edu.tusur.ru/publications/8515 (дата обращения: 06.08.2018).

12.3.2. Учебно-методические пособия для лиц с ограниченными возможностями здоровья и инвалидов

Учебно-методические материалы для самостоятельной и аудиторной работы обучающихся из числа лиц с ограниченными возможностями здоровья и инвалидов предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации.

Для лиц с нарушениями зрения:

- в форме электронного документа;
- в печатной форме увеличенным шрифтом.

Для лиц с нарушениями слуха:

- в форме электронного документа;
- в печатной форме.

Для лиц с нарушениями опорно-двигательного аппарата:

- в форме электронного документа;
- в печатной форме.

12.4. Профессиональные базы данных и информационные справочные системы

1. Образовательный портал университета (http://edu.tusur.ru).
2. При изучении дисциплины рекомендуется обращаться к базам данных, информационно-справочным и поисковым системам, к которым у ТУСУРа открыт доступ: https://lib.tusur.ru/ru/resursy/bazy-dannyh

13. Материально-техническое обеспечение дисциплины и требуемое программное обеспечение

13.1. Общие требования к материально-техническому и программному обеспечению дисциплины

13.1.1. Материально-техническое и программное обеспечение для лекционных занятий

Для проведения занятий лекционного типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации используется учебная аудитория с количеством посадочных мест не менее 22-24, оборудованная доской и стандартной учебной мебелью. Имеются демонстрационное оборудование и учебно-наглядные пособия, обеспечивающие тематические иллюстрации по лекционным разделам дисциплины.

13.1.2. Материально-техническое и программное обеспечение для лабораторных работ

Лаборатория «Информатика и программирование»

учебная аудитория для проведения занятий практического типа, учебная аудитория для про-

ведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 428 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core 2 Duo E6550 2.3 ГГц, ОЗУ – 2 Гб, жесткий диск – 250 Гб (14 шт.);
- Меловая доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

– Git 2.11.03, GNU GPLv2
– Google Chrome
– Microsoft Visual Studio 2015
– Microsoft Windows 7 Pro

Лаборатория «Муниципальная информатика»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 432б ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i5-2320 3.0 ГГц, ОЗУ – 4 Гб, жесткий диск – 500 Гб (12 шт.);
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

– Git 2.11.03, GNU GPLv2
– Google Chrome
– Microsoft Visual Studio 2015
– Microsoft Windows 10 Pro

Лаборатория «Распределенные вычислительные системы»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 432а ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i5-3330 3.0 ГГц, ОЗУ – 4 Гб, жесткий диск – 500 Гб (12 шт.);
- Меловая доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

– Git 2.11.03, GNU GPLv2
– Google Chrome
– Microsoft Visual Studio 2015
– Microsoft Windows 10 Pro

Лаборатория «Операционные системы и СУБД»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 430 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core 2 Duo E6550 2.3 ГГц, ОЗУ – 2 Гб, жесткий диск – 250 Гб (12 шт.);
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- Git 2.11.03, GNU GPLv2
- Google Chrome
- Microsoft Visual Studio 2015
- Microsoft Windows 7 Pro

Лаборатория «Программная инженерия»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 409 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i3-6300 3.2 ГГц, ОЗУ – 8 Гб, жесткий диск – 500 Гб (10 шт.);
- Проектор Optoma Ex632.DLP;
- Экран для проектора Lumian Mas+Er;
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- GCC, GNU GPLv3
- Google Chrome
- Microsoft Visual Studio 2015

13.1.3. Материально-техническое и программное обеспечение для самостоятельной работы

Для самостоятельной работы используются учебные аудитории (компьютерные классы), расположенные по адресам:

- 634050, Томская область, г. Томск, Ленина проспект, д. 40, 233 ауд.;
- 634045, Томская область, г. Томск, ул. Красноармейская, д. 146, 201 ауд.;
- 634034, Томская область, г. Томск, Вершинина улица, д. 47, 126 ауд.;
- 634034, Томская область, г. Томск, Вершинина улица, д. 74, 207 ауд.

Состав оборудования:

- учебная мебель;
- компьютеры класса не ниже ПЭВМ INTEL Celeron D336 2.8ГГц. - 5 шт.;
- компьютеры подключены к сети «Интернет» и обеспечивают доступ в электронную информационно-образовательную среду университета.

Перечень программного обеспечения:

- Microsoft Windows;
- OpenOffice;
- Kaspersky Endpoint Security 10 для Windows;
- 7-Zip;
- Google Chrome.

13.2. Материально-техническое обеспечение дисциплины для лиц с ограниченными возможностями здоровья и инвалидов

Освоение дисциплины лицами с ограниченными возможностями здоровья и инвалидами осуществляется с использованием средств обучения общего и специального назначения.

При занятиях с обучающимися **с нарушениями слуха** предусмотрено использование звукоусиливающей аппаратуры, мультимедийных средств и других технических средств приема/передачи учебной информации в доступных формах, мобильной системы преподавания для обучающихся с инвалидностью, портативной индукционной системы. Учебная аудитория, в которой занимаются обучающиеся с нарушением слуха, оборудована компьютерной техникой, аудиотехникой, видеотехникой, электронной доской, мультимедийной системой.

При занятиях с обучающимися **с нарушениями зрениями** предусмотрено использование в лекционных и учебных аудиториях возможности просмотра удаленных объектов (например, текста на доске или слайда на экране) при помощи видеоувеличителей для комфортного просмотра.

При занятиях с обучающимися **с нарушениями опорно-двигательного аппарата** используются альтернативные устройства ввода информации и другие технические средства приема/передачи учебной информации в доступных формах, мобильной системы обучения для людей с инвалидностью.

14. Оценочные материалы и методические рекомендации по организации изучения дисциплины

14.1. Содержание оценочных материалов и методические рекомендации

Для оценки степени сформированности и уровня освоения закрепленных за дисциплиной компетенций используются оценочные материалы в составе:

14.1.1. Тестовые задания

1. Какую оптимизацию кода программного обеспечения следует делать "по умолчанию" (в первую очередь)?

1) С точки зрения занимаемой машинной памяти
2) С точки зрения читаемости кода и простоты понимания его структуры
3) С точки зрения количества строк кода
4) С точки зрения скорости работы программы

2. Где будут доступны для использования члены класса, помеченные как protected ?

1) Только внутри кода самого класса
2) Внутри кода самого класса и из кода классов наследников
3) Только из внешнего кода
4) Только внутри классов потомках

3. Дан следующий класс (псевдокод):
Class Number:
Public:
Value;
Number(V): Value = V;
SetValue(V): Value = V;
GetValue(): return Value;
Есть ли подозрение на ошибку в таком описании и если да, то на какую?

1) В описании класса нет никакой ошибки, все верно
2) Переменная класса Value должна бы быть в защищенной области, а не в публичной
3) Имена используемых функций запрещены в большинстве языков программирования
4) В конструкторе должна выделяться память под переменную, а этого не сделано

4. Какой вариант наследования выглядит правильным (Потомок <- Предок)?

1) Медведь_Копатыч <- Медведь <- Животное
2) Медведь <- Медведь_Копатыч <- Животное
3) Животное <- Медведь <- Медведь_Копатыч
4) Медведь_Копатыч <- Животное <- Медведь

5. Какое утверждение относительное строчки кода:
A = B C;
в языке C+является правильным?

1) Это сложение двух чисел
2) Это конкатенация двух строк
3) Так как в языке C+имеется перегрузка операторов, то без информации о типе B и C, невозможно сказать какую операцию выполняет данная строчка
4) Это побитовое логическое сложение двух целых чисел

6. При написании программы на C++, которая обрабатывает очень большие массивы данных из комплексных чисел нужно выбрать вариант их определения. Для того, чтобы оптимизировать программу с точки зрения используемой памяти и не потерять читаемость кода, какой из указанных ниже вариантов скорее всего подходит лучше других?

1) Класс, в котором действительная и мнимая часть являются отдельными членами
2) Массив из двух чисел, в одном находится действительная часть, в другом мнимая
3) Два больших массива одинаковой величины, один из которых хранит действительную часть, другой мнимую
4) Структура из двух чисел (действительная и мнимая часть), обернутая в конструкцию typedef

7. Какими особенностями обладает класс, реализующий шаблон проектирования “Одиночка (Singleton)” ?

1) Присутствует только один конструктор
2) Нет публичного конструктора, но есть метод возвращающий ссылку или указатель на объект класса
3) Реализуется специальной директивой компилятора
4) Отсутствуют приватные данные

8. Для чего нужны исключения (exception)?

1) Для обработки ошибок, возникающих в ходе выполнения программы
2) Для исключения участков кода из программы
3) Для обработки аппаратных прерываний
4) Чтобы отслеживать несанкционированную активность других (по отношению к вашей) программ

9. Мы решили реализовать кроссплатформенную реализацию интерфейса используя шаблон проектирования "Фабричный метод". Ниже приведены классы, их ответственность и взаимосвязи. Какой из вариантов является верным?

1) Классы winButton и linuxButton реализуют одни и теже методы для разных ОС.
Логика нашего приложения написана в классе Dialog, который использует классы winButton и linuxButton для реализации своей бизнес-логики, при этом использование того или иного класса реализуется с помощью условных операторов.
В классе Dialog используется фабричный метод для создания внутренних объектов типа Button опять же с использованием условных операторов.

2) Класс Button является полностью абстрактным. От него наследуются классы winButton и linuxButton, которые реализуют все методы класса предка.
Логика нашего приложения написана в классе Dialog, который использует класс Button для реализации своей бизнес-логики. В классе Dialog используется фабричный метод для создания внутренних объектов типа Button. Этот метод знает под какую ОС нужно создавать Button.
Таким образом логика работа диалога не зависит от ОС (она находится в конкретных методах класса Dialog), и этот код переиспользуется для всех ОС.
А логика создания Button в зависимости от типа ОС находится внутри класса Dialog в его фабричном методе создания.

3) Класс Button является полностью абстрактным. От него наследуются классы winButton и linuxButton, которые реализуют все методы класса предка.
Логика нашего приложения написана в классе Dialog, который использует класс Button для реализации своей бизнес-логики. В классе Dialog используется абстрактный фабричный метод для создания внутренних объектов типа Button.
От Класса Dialog наследуются конкретные классы для каждой из операционных систем, и в каждом из этих классов реализуется метод создания элемента Button в зависимости от типа ОС.
Таким образом логика работы диалога не зависит от ОС (она описана в классе предке), и этот код переиспользуется для всех ОС (всех потомков) класса Диалог.

4) Класс Button является полностью абстрактным. От него наследуются классы winButton и linuxButton, которые реализуют все методы класса предка.
В классе Dialog используется абстрактный фабричный метод для создания внутренних объектов типа Button. Этот метод знает под какую ОС нужно создавать Button.
От Класса Dialog наследуются конкретные классы для каждой из операционных систем, и в каждом из этих классов реализуется бизнес логика работы Dialog.

10. Имеется ранее написанный код, который мы хотим дополнить новыми, уже готовыми классами. Проблема заключается в том, что эти новые классы имеют интерфейс, несовместимый с нашим кодом, но при этом реализуют всю нужную нам дополнительную функциональность. Какой из вариантов решения проблемы имеет смысл рассматривать в первую очередь?

1) Переписать все части нашей программы, так, чтобы она работала с новыми классами напрямую
2) Переписать новые классы так, чтобы они соответствовали старому коду
3) Обернуть новые классы, используя шаблон "Адаптер" так, чтобы интерфейсы соответствовали остальному коду программы и потом встроить их
4) Отказаться от решения проблемы

11. Какой шаблон проектирования предназначен для организации поведения "отмены пре-

дыдущего действия" (например, в текстовом редакторе)?

1) Команда совместно с Хранитель
2) Цепочка обязанностей
3) Состояние
4) Посредник

12. У нас есть большой класс, реализующий сразу и логику управляющих воздействий, и логику их исполнения.

Псевдокод:

```
Class SuperImplemantation:
ClickOnTurnOnMouse() {
if State is TV then TurnOnTV()
if state is Radio then TurnOnRadio()
}
ClickOnTurnOnKeyboard() {
if State is TV then TurnOnTV()
if state is Radio then TurnOnRadio()
}
ClickOnVolumeIncreaseOnKeyboard() {
IncreaseVolume( 10 );
}
ClickOnVolumeIncreaseOnMouse() {
IncreaseVolume( 3 );
}
TurnOnTV() {...}
TurnONRadio() {...}
IncreaseVolume( value ) {...}
```

Мы решили разделить работу с интерфейсом и бизнес логику используя шаблон проектирования "Мост". Какое разделение выглядит наиболее логичным и правильным с точки зрения этого паттерна?

1) Делаем четыре класса, каждый из которых является комбинацией UI и Device, а клиент будет использовать нужный ему класс:

```
Class: UIMouseDeviceTV:
TurnON()
VolumeIncrease()
Class: UIMouseDeviceRadio:
TurnON()
VolumeIncrease()
Class: UIKeyboardDeviceTV:
TurnON()
VolumeIncrease()
Class: UIKeyboardDeviceRadio:
TurnON()
VolumeIncrease()
```

2) Делаем два абстрактных класса для организации паттерна. Делаем два класса, в которых реализуем реакцию на интерфейс и два класса имплементации, в которых реализуем логику, а клиенту даем возможность связать их в любой конфигурации:

```
Class UI:
UI( Device ) {self.UI = UI}
Class UIKeyboard: UI
TurnON()
VolumeIncrease()
private: Device
Class UIMouse: UI
TurnON()
VolumeIncrease()
private: Device
Class Device: ...
Class TV: Device
TurnON()
IncreaseVolume()
Class Radio: Device
TurnON()
IncreaseVolume()
```

3) Реализуем один абстрактный класс, от которого наследуем четыре класса реализующих интерфейс и логику:

```
Class UI_Device:...

Class: UIMouseDeviceTV: UI_Device
TurnON()
VolumeIncrease()
Class: UIMouseDeviceRadio: UI_Device
TurnON()
VolumeIncrease()
Class: UIKeyboardDeviceTV: UI_Device
TurnON()
VolumeIncrease()
Class: UIKeyboardDeviceRadio: UI_Device
TurnON()
VolumeIncrease()
```

4) Оставляем всю логику интерфейса в одном классе, а реализацию логики разбиваем на два класса:

```
Class UI:
private: Device
private: State
UI( state, device ) { self.State = state, self.Device = device }
ClickOnTurnOnMouse() {
if State is TV then (TV)Device.TurnOn()
if state is Radio then (Radio)Device.TurnOn()
}
ClickOnTurnOnKeyboard() {
```

```
if State is TV then (TV)Device.TurnOn()
if state is Radio then (Radio)Device.TurnOn()
}
ClickOnVolumeIncreaseOnKeyboard() {
if state is TV then (TV)Device.IncreaseVolume( 10 );
if state is Radio then (Radio)Device.IncreaseVolume( 10 );
}
ClickOnVolumeIncreaseOnMouse() {
if state is TV then (TV)Device.IncreaseVolume( 3 );
if state is Radio then (Radio)Device.IncreaseVolume( 3 );
}
Class Device ...
Class TV: Device
TurnOn() {...}
IncreaseVolume() {...}
Class Radio: Device
TurnOn() {...}
IncreaseVolume() {...}
```

13. В каком из указанных случаев применим шаблон проектирования "Фасад"?

1) Когда Ваш код должен создавать разные представления какого-то объекта
2) Когда Вам нужно представить простой или урезанный интерфейс к сложной подсистеме
3) Когда у Вас есть объект, поведение которого кардинально меняется в зависимости от внутреннего состояния, причём типов состояний много, и их код часто меняется
4) Когда код класса содержит множество больших, похожих друг на друга, условных операторов, которые выбирают поведения в зависимости от текущих значений полей класса

14. Для оценки качества исходного кода программ существует несколько принципов, известных как SOLID. В них же определены признаки плохого проекта. Какой из указанных ниже признаков не является показателем плохого проекта?

1) Система с трудом поддается изменениям, поскольку любое минимальное изменение вызывает эффект "снежного кома", затрагивающего другие компоненты системы.
2) В системе используется принцип "единственной ответственности", т.е. один класс отвечает лишь за одну функцию системы и его изменение может потребоваться только в случае изменения требований к одной функции
3) В результате осуществляемых изменений система разрушается в тех местах, которые не имеют прямого отношения к непосредственно изменяемому компоненту.
4) Достаточно трудно разделить систему на компоненты, которые могли бы повторно использоваться в других системах.

15. Какая из приведенных ниже формулировок является правильной для "принципа открытости/закрытости" (ООП, принципы SOLID)?

1) Все переменные неизменяемые внутри функций должны быть объявлены как константы
2) Программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
3) При разработке классов следует предусматривать меры, которые не позволят его пользователям копировать объекты этих классов

4) Следует использовать вызов исключений при обнаружении попытки прямого доступа к защищенным членам классов

16. Какая из нижеприведенных формулировок является правильной для "принципа подстановки Барбары Лисков" (ООП, принципы SOLID)?

1) При попытке подстановки в аргумент функции неправильного типа компилятор должен выдавать ошибку
2) Механизм перегрузки функций должен позволять компилятору самому делать выбор о том, какую функцию выбрать в зависимости от типа переданных аргументов
3) Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом
4) Запрещена подстановка в параметры функций классов потомков к определенному в описании функции классу

17. Какой из нижеперечисленных способов разработки может улучшить UI для вашей программы и ускорить саму разработку?

1) UI вообще не нужно проектировать, как получится, так и получится, т.к. это ускорит разработку логики программы
2) Следует сначала проработать всю бизнес-логику, потом пригласить профессионального создателя UI интерфейсов, который все сделает, основываясь на указанной бизнес-логике
3) Имеет смысл применить какую-либо из упрощенных методик проектирования UI для разработки прототипа и вести разработку основываясь на нем
4) Нужно надавить на заказчика и заявить ему, что он должен предоставить полностью разработанный и продуманный интерфейс, а вы уже будете его реализовывать

18. При конструировании ПО мы решили применить стороннюю библиотеку. Какой набор вопросов нужно разрешить, чтобы применить ее?

1) Компилируемость библиотеки для наших нужд, качественная работа библиотеки
2) Компилируемость библиотеки для наших нужд, качественная работа библиотеки, соответсвие лицензии библиотеки нашему ПО, объем исходного кода библиотеки
3) Соответствие лицензии библиотеки и нашему ПО, компилируемость библиотеки для наших нужд, качественная работа библиотеки
4) Компилируемость библиотеки для наших нужд, качественная работа библиотеки, соответствие лицензии библиотеки нашему ПО, объем, который эта библиотека привнесет в дистрибутив нашего продукта

19. При разработке ПО для ОС семейства Windows нам нужно решить задачу обновления ПО на новые версии. Какой из способов выглядит наиболее простым и надежным для использования?

1) Нужно при установке ПО требовать e-mail пользователя, и при выходе обновлений ПО рассылать уведомление, оставляя ответственность за обновление на совести пользователя
2) Нужно сделать ВЕБ сервис, при опросе которого наше ПО сможет узнавать, что существует более новая версия и она будет выдавать пользователю сообщение, оставляя обновление на его совести
3) Нужно воспользоваться сервисом, предоставляемым Microsoft по распространению ПО
4) Нужно попробовать сделать скрытый установщик, который без ведома пользователя и по

максимуму скрытно от ОС скачает и установит обновления

20. Какую из нижеперечисленных функций неспособна выполнять ни одна система контроля версий?

1) Хранение разных версий исходного кода
2) Организация работы нескольких разработчиков
3) Контроль за правильностью исходного кода
4) Хранение резервных копий исходного кода

14.1.2. Темы контрольных работ

Шаблоны проектирования

14.1.3. Зачёт

Расскажите принцип действия и внутреннее устройство шаблона проектирования "Синглтон"

Расскажите принцип действия и внутреннее устройство шаблона проектирования "Декоратор"

Расскажите принцип действия и внутреннее устройство шаблона проектирования "Фабрика"

Расскажите принцип действия и внутреннее устройство шаблона проектирования "Представитель"

Расскажите принцип действия и внутреннее устройство шаблона проектирования "Интерфейс"

14.1.4. Темы лабораторных работ

Конструирование бизнес-логики и архитектуры простого приложения с применением объектно-ориентированного проектирования.

Применение шаблонов проектирования программного обеспечения.

14.1.5. Методические рекомендации

Темы для самостоятельного изучения:

Основы объектно-ориентированного программирования и проектирования.

Шаблоны ООП.

Чистая архитектура и принципы SOLID.

14.2. Требования к оценочным материалам для лиц с ограниченными возможностями здоровья и инвалидов

Для лиц с ограниченными возможностями здоровья и инвалидов предусмотрены дополнительные оценочные материалы, перечень которых указан в таблице 14.

Таблица 14 – Дополнительные материалы оценивания для лиц с ограниченными возможностями здоровья и инвалидов

Категории обучающихся	Виды дополнительных оценочных материалов	Формы контроля и оценки результатов обучения
С нарушениями слуха	Тесты, письменные самостоятельные работы, вопросы к зачету, контрольные работы	Преимущественно письменная проверка
С нарушениями зрения	Собеседование по вопросам к зачету, опрос по терминам	Преимущественно устная проверка (индивидуально)
С нарушениями опорно-двигательного аппарата	Решение дистанционных тестов, контрольные работы, письменные самостоятельные работы, вопросы к зачету	Преимущественно дистанционными методами
С ограничениями по общемедицинским	Тесты, письменные самостоятельные работы, вопросы к зачету,	Преимущественно проверка методами исходя из состояния

показаниям	контрольные работы, устные ответы	обучающегося на момент проверки

14.3. Методические рекомендации по оценочным материалам для лиц с ограниченными возможностями здоровья и инвалидов

Для лиц с ограниченными возможностями здоровья и инвалидов предусматривается доступная форма предоставления заданий оценочных средств, а именно:

- в печатной форме;
- в печатной форме с увеличенным шрифтом;
- в форме электронного документа;
- методом чтения ассистентом задания вслух;
- предоставление задания с использованием сурдоперевода.

Лицам с ограниченными возможностями здоровья и инвалидам увеличивается время на подготовку ответов на контрольные вопросы. Для таких обучающихся предусматривается доступная форма предоставления ответов на задания, а именно:

- письменно на бумаге;
- набор ответов на компьютере;
- набор ответов с использованием услуг ассистента;
- представление ответов устно.

Процедура оценивания результатов обучения лиц с ограниченными возможностями здоровья и инвалидов по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

- в форме электронного документа;
- в печатной форме увеличенным шрифтом.

Для лиц с нарушениями слуха:

- в форме электронного документа;
- в печатной форме.

Для лиц с нарушениями опорно-двигательного аппарата:

- в форме электронного документа;
- в печатной форме.

При необходимости для лиц с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения может проводиться в несколько этапов.