ПРИЛОЖЕНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ В.Ф. УТКИНА»

Кафедра «Вычислительной и прикладной математики»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ
«Конструирование ПО»

Направление подготовки

09.03.04 Программная инженерия

Направленность (профиль) подготовки

«Программная инженерия»

Квалификация выпускника — бакалавр

Форма обучения — очная

Рязань

1. ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов (контрольных заданий, описаний форм и процедур), предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной профессиональной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций, обучающихся целям и требованиям ОПОП.

Основная задача – обеспечить оценку уровня сформированности общекультурных, общепрофессиональных и профессиональных компетенций.

Контроль знаний обучающихся проводится в форме промежуточной аттестации.

Промежуточный контроль по дисциплине осуществляется проведением экзамена/зачета/курсовой работы.

2. ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Сформированность каждой компетенции в рамках освоения данной дисциплины оценивается по трехуровневой шкале:

1) пороговый уровень является обязательным для всех обучающихся по завершении освоения дисциплины;

2) продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;

3) эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

Уровень освоения компетенций, формируемых дисциплиной:

а) описание критериев и шкалы оценивания тестирования:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 85 до 100%
2 балла (продвинутый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 75 до 84%
1 балл (пороговый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 60 до 74%
0 баллов	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 0 до 59%

б) описание критериев и шкалы оценивания теоретического вопроса:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	выставляется студенту, который дал полный ответ на вопрос, показал глубокие систематизированные знания, смог привести примеры, ответил на дополнительные вопросы преподавателя.
2 балла (продвинутый уровень)	выставляется студенту, который дал полный ответ на вопрос, но на некоторые дополнительные вопросы преподавателя ответил только с помощью наводящих вопросов.
1 балл (пороговый уровень)	выставляется студенту, который дал неполный ответ на вопрос в билете и смог ответить на дополнительные вопросы только с помощью преподавателя.
0 баллов	выставляется студенту, который не смог ответить на вопрос

в) описание критериев и шкалы оценивания практического задания:

Шкала оценивания	Критерий
3 балла (эталонный уровень)	Задача решена верно
2 балла (продвинутый уровень)	Задача решена верно, но имеются технические неточности в расчетах
1 балл (пороговый уровень)	Задача решена верно, с дополнительными наводящими вопросами преподавателя
0 баллов	Задача не решена

На экзамен выносится: тестовое задание, 1 практическое задание и 1 теоретический вопрос. Студент может набрать максимум 9 баллов. Итоговый суммарный балл студента, полученный при прохождении промежуточной аттестации, переводится в традиционную форму по системе «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Шкала оценивания	Критерий	
отлично (эталонный уровень)	8 – 9 баллов	Обязательным условием является выполнение всех предусмотренных в течение семестра заданий
хорошо (продвинутый уровень)	6 – 7 баллов	
удовлетворительно (пороговый уровень)	4 – 5 баллов	
неудовлетворительно	0 – 3 баллов	Студент не выполнил всех предусмотренных в течение семестра текущих заданий

3. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

№ п/п	Контролируемые разделы (темы) дисциплины (результаты по разделам)	Код контролируемой компетенции (или её части)	Наименование оценочного мероприятия
1	2	3	4
1	Тема 1. Знакомство со средами разработки на примере Microsoft Visual Studio	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
2	Тема 2. Изучение важности использования форматирования и стиля при написании кода, знакомство соглашением по написанию кода.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
3	Тема 3. Изучение объектно-ориентированного программирования, знакомство со структурным программированием.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
4	Тема 4. Изучение основ работы с системами контроля версий. Знакомство с системами контроля ошибок. Основная литература	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
5	Тема 5. Изучение порождающих шаблонов проектирования. Освоение их использования на лабораторных работах.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
6	Тема 6. Изучение структурных шаблонов	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен

	проектирования. Освоение их использования в курсовой работе.		
7	Тема 7. Изучение поведенческих шаблонов проектирования. Освоение их использования на лабораторных работах	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Экзамен
8	Тема 8. Модульные тесты, особенности разработки. Разработка основанная на тестах	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
9	Тема 9. Делегаты, лямбда выражения. Событийный подход к разработке программного обеспечения.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
10	Тема 10. Основы параллельной обработки данных. Потоки. Синхронизация потоков.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
11	Тема 11. Основы совершенствования кода без видимых изменений для пользователя.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
12	Тема 12. Основы разработки интерфейса пользователя. Дружественный интерфейс.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
13	Тема 13. Основы работы с графикой. Отображение текста. 2D преобразования.	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет
14	Тема 14. Курсовая работа	ПК-1.1 ПК-1.2 ПК-1.3 ПК-2.1 ПК-2.2	Зачет/Экзамен

4. ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ

4.1. Промежуточная аттестация (экзамен)

ПК-1 Способен разрабатывать требования, проектировать и выполнять программную реализацию программного обеспечения
ПК-1.1 Анализирует требования к программному обеспечению
ПК-1.2 Разрабатывает технические спецификации на программные компоненты

а) типовые тестовые вопросы закрытого типа:

1. Требования к программному обеспечению разрабатывает:
   - Заказчик
   - ***Аналитик***
   - Разработчик
   - Тестировщик
2. Требования к программному обеспечению подготавливаются:
   - После разработки ПО
   - Во время разработки ПО
   - **Перед началом разработки ПО**
   - Для разработки ПО требования не обязательны
3. Что не содержится в системных требованях к ПО
   - **Смета затрат на разработку**
   - Графические материалы
   - UML диаграммы
   - Нефункциональные требования
4. UML диаграмма активности это
   - Представление состояний объектов и связей между ними
   - **Представление последовательности действий, ветвлений в виде дерева**
   - Представление последовательности шагов для достижения конечного результата
   - Представление классов, отношений и связей между ними

5. UML диаграмма последовательности это
    - Представление состояний объектов и связей между ними
    - Представление последовательности действий, ветвлений в виде дерева
    - **Представление последовательности шагов для достижения конечного результата**
    - Представление классов, отношений и связей между ними
6. UML диаграмма состояний это
    - **Представление состояний объектов и связей между ними**
    - Представление последовательности действий, ветвлений в виде дерева
    - Представление последовательности шагов для достижения конечного результата
    - Представление классов, отношений и связей между ними
7. UML диаграмма классов это
    - Представление состояний объектов и связей между ними
    - Представление последовательности действий, ветвлений в виде дерева
    - Представление последовательности шагов для достижения конечного результата
    - **Представление классов, отношений и связей между ними**
8. Функциональные требования это
    - Требования к интерфейсу разрабатываемой программы
    - Требования к написанию функций в программе
    - **Требования к разрабатываемому функционалу, включающему ожидания пользователя**
    - Требования написанные функциональным языком
9. Нефункциональные требования это
    - Требования к интерфейсу разрабатываемой программы
    - **Требования не связанные напрямую с достижением целей и удовлетворением последовательности пользователя**
    - Требования к разрабатываемому функционалу, включающему ожидания пользователя
    - Требования написанные нефункциональным языком

б) типовые тестовые вопросы открытого типа:

Дайте определение водопадной модели разработки ПО: это методология разработки программного обеспечения, основанная на поэтапном создании итераций, каждая из которых включает в себя следующие этапы:

1. Определение требований
2. Проектирование
3. Разработка
4. Тестирование
5. Внедрение
6. Поддержка

Дайте определение спиральной модели разработки ПО: это методология разработки программного обеспечения основанная на циклической разработке, включающей этапы:

1. Планирование
2. Проектирование
3. Анализ
4. Разработка
5. Тестирование
6. Эксплуатация
6. Обновление

ОПК-6: Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;

ПК-1 Способен разрабатывать требования, проектировать и выполнять программную реализацию программного обеспечения
ПК-1.3 Проектирует программное обеспечение и выполняет его программную реализацию

1. ***Выбери правильные варианты определения класса***
   - ***Задаёт общее поведение для объектов***
   - ***Описывается на этапе написания кода***
   - ***Создаётся в процессе работы программы***
   - ***Должен наследоваться от другого класса***
2. ***Конструктор это***
   - ***Интерфейс для проектирования классов***
   - ***Ключевое слово, используемое при разработке***
   - ***Специальный метод класса***
   - ***Набор для разработки***
3. ***Виртуальный метод***
   - ***Можно переопределять***
   - ***Обязательно нужно переопределять***
   - ***Метод, который не возвращает значения***
   - ***Метод без параметров***
4. ***Абстрактный метод***
   - ***Один для всей иерархии классов***
   - ***Не имеет реализации***
   - ***Метод, который не возвращает значения***
   - ***Метод без параметров***
5. ***Статический метод***
   - ***Не имеет реализации***
   - ***Метод, который не возвращает значения***
   - ***Один для всей иерархии классов***
   - ***Метод без параметров***
6. ***Абстрактный класс***
   - ***Нельзя создать объект данного класса***
   - ***Имеет хотя бы один абстрактный метод***
   - ***Не может иметь наследников***
   - ***Должен быть базовым классом***

б) типовые тестовые вопросы открытого типа:

1. Опишите клиент-серверную архитектуру ПО

Архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. Типичным, упрощённым, примером реализации архитектуры клиент-сервер можно назвать различные Web сайты, которые содержат страницы с данными, а также логику по их отображения. Клиенты передают на сайты запросы на предоставление информации, а в ответ получают содержимое запрошенных страниц.

2. Опишите компонентную архитектуру ПО:

В компонентной архитектуре основное внимание уделяется разложению дизайна на отдельные функциональные или логические компоненты. Обычно в приложениях используются компоненты пользовательского интерфейса (их часто называют элементами управления), такие как таблицы и кнопки, а также вспомогательные или служебные компоненты, предоставляющие определенный набор функций, используемых в других компонентах.

3. Опишите многослойную (многоуровневую) архитектуру ПО:

Многоуровневая архитектура обеспечивает группировку связанной функциональности приложения в разных слоях, выстраиваемых вертикально, поверх друг друга. Функциональность каждого слоя объединена общей ролью или ответственностью. Слои приложения могут размещаться физически на одном компьютере (на одном уровне) или быть распределены по

разным компьютерам (n-уровней), и связь между компонентами разных уровней осуществляется через строго определенные интерфейсы. Например, типовое Веб-приложение состоит из слоя представления, слоя логики и слоя данных.

4. Опишите архитектуру, основанную на шине сообщений:

Основанная на шине сообщений архитектура описывает принцип использования программной системы, которая может принимать и отправлять сообщения по одному или более каналам связи, обеспечивая, таким образом, приложениям возможность взаимодействия без необходимости знания конкретных деталей друг о друге.
Например: часть клиентов работает через Web сервер с корпоративными сервисами, каждый из которых установлен на отдельный сервер:

- Получение данных из базы данных;
- Работа с почтой;
- Работа с файлами.

Также доступ к сервисам напрямую имеют клиенты, которые, например, подключены к корпоративной сети компании. Весь обмен информации производится через шину данных. Программы отсылают в ней запросы, шина данных определяет получателя (сервер/сервис, к которому адресован запрос) и направляет его к нему, а затем возвращает результат отправителю.

5. Опишите объектно-ориентированную архитектуру ПО:

Объектно-ориентированная архитектура – это парадигма проектирования, основанная на разделении ответственностей приложения или системы на самостоятельные пригодные для повторного использования объекты, каждый из которых содержит данные и поведение, относящиеся к этому объекту. При объектно-ориентированном проектировании система рассматривается не как набор подпрограмм и процедурных команд, а как наборы взаимодействующих объектов. Объекты обособлены, независимы и слабо связаны; обмен данными между ними происходит через интерфейсы путем вызова методов и свойств других объектов и отправки/приема сообщений.

6. Опишите сервисно-ориентированную архитектуру ПО:

Сервисно-ориентированная архитектура обеспечивает возможность предоставлять функциональность приложения в виде набора сервисов и создавать приложения, использующие программные сервисы. Сервисы слабо связаны, потому что используют основанные на стандартах интерфейсы, которые могут быть вызваны, опубликованы и обнаружены. Данная архитектура может обеспечить упаковку данных в сервисы, поддерживающие возможность взаимодействия и использующие для передачи информации широкий диапазон протоколов и форматов данных.
В настоящее время практически все современные языки программирования поддерживают формат REST/SOAP сервисов, которые позволяют различным системам обмениваться информацией вне зависимости от языка программирования, на котором она была написана.

7. Опишите, что такое компилятор и укажите принцип его работы.

Компилятор – это программа, которая преобразует исходный код программы, написанной на одном языке программирования (называемом исходным языком), в эквивалентный код на другом языке (называемом целевым языком), который может быть выполнен на целевой платформе (например, компьютере или микроконтроллере).
Основная задача компилятора – это перевести исходный код программы в машинный код или в код на низкоуровневом языке, который может быть понятен компьютеру. Этот процесс включает лексический анализ (разбиение исходного кода на лексемы), синтаксический анализ (построение дерева синтаксического разбора), семантический анализ (проверка правильности использования языковых конструкций) и генерацию кода (создание эквивалентного кода на целевом языке).

8. Опишите, что такое интерпретатор и укажите принцип его работы.

Интерпретатор – это программа, которая выполняет исходный код программы построчно, без предварительной компиляции. Он читает и анализирует каждую строку исходного кода программы и непосредственно выполняет соответствующие инструкции.
В отличие от компилятора, который преобразует весь исходный код программы в машинный код или в код на низкоуровневом языке, интерпретатор выполняет программу "на лету". Он обычно работает с высокоуровневыми языками программирования, такими как Python, JavaScript или Ruby.

ПК-2 Способен выполнять проектирование программных систем среднего и крупного масштаба сложности

ПК-2.1 Разрабатывает бизнес-требования к программной системе

1. Выбери правильные варианты определения бизнес-требования
   - **Условие или возможность, необходимые причастному лицу для решения проблемы или достижения цели**
   - Подлежащее выполнению условие или способность, которой должно обладать решение или его компонент для соответствия контракту, стандарту, спецификации или другому формальному документу
   - Описание выгоды, которую получит заказчик после разработки требуемого продукта
2. Кто разрабатывает бизнес-требования
   - Системный аналитик
   - **Бизнес-аналитик**
   - Технический писатель
   - Заказчик
3. Выберите вид бизнес требования «значимые характеристики продуктов и услуг»:
   - Сотрудники должны быть своевременно проинформированы о назначенных им задачах
   - Решение о выдаче кредита должно приниматься на основании…
   - С момента обнаружения дефекта и до его полного устранения должна регистрироваться и храниться следующая информация
   - Покупатели должны иметь возможность оформить и оплатить покупку самостоятельно посредством мобильного приложения
   - **Клиент должен получить пиццу горячей и не умереть ожидая ее**
4. Выберите вид бизнес-требования «Распознавание и обработка событий»:
   - Очередь на кассе должна быть не длиннее 4 покупателей
   - **Банк должен своевременно получать оповещения о сбоях в работе банкоматов**
   - Решение о выдаче кредита должно приниматься на основании…
   - Для принятия решений об участии в тендере необходимы результаты предварительной оценки себестоимости проекта
5. Выберите вид бизнес-требования «Принятие решений»:
   - Оформление предварительно одобренного кредита должно завершаться за одно посещение клиента
   - К моменту начала этапа работ, все нужные материалы и оборудование должны находиться на объекте строительства и быть готовыми к использованию
   - **По каждому обнаруженному дефекту, на основе имеющейся информации, должно быть оперативно принято решение о способе его устранения**
   - Клиент должен иметь возможность отказаться от заказа до момента его отправки
6. Выберите вид бизнес-требования «Сбор, обработка, хранение и предоставление информации»:
   - Очередь на кассе должна быть не длиннее 4 покупателей
   - Сотрудники должны быть своевременно проинформированы о назначенных им задачах
   - Отпускная цена на товар для конкретного клиента определяется на основании… в зависимости от …
   - **Для принятия решений об участии в тендере необходимы результаты предварительной оценки себестоимости проекта**
7. Выберите вид бизнес-требования «Обеспечение возможности выполнения действий»:
   - **В случае неудовлетворенности качеством обслуживания, клиент должен иметь возможность направить жалобу по телефону, по электронной почте или через сайт компании**

- Оформление предварительно одобренного кредита должно завершаться за одно посещение клиента
- К моменту начала этапа работ, все нужные материалы и оборудование должны находиться на объекте строительства и быть готовыми к использованию
- По каждому обнаруженному дефекту, на основе имеющейся информации, должно быть оперативно принято решение о способе его устранения

8. Выберите вид бизнес-требования «Предотвращение возможности выполнения действий»:
   - Очередь на кассе должна быть не длиннее 4 покупателей
   - **Должна исключаться возможность доступа к защищенным ресурсам лиц, не являющихся сотрудниками компании**
   - Банк должен своевременно получать оповещения о сбоях в работе банкоматов
   - Решение о выдаче кредита должно приниматься на основании…
   - Для принятия решений об участии в тендере необходимы результаты предварительной оценки себестоимости проекта
9. Какой раздел бизнес-требования описывает цели и задачи проекта?
   - **Введение**
   - Функционал
   - Ограничения
   - Архитектура
10. Что из перечисленного НЕ является этапом процесса написания бизнес-требований?
    - Определение требований
    - Формирование требований
    - **Анализ потребностей**
    - Внедрение требований
11. Что такое “профиль пользователя” в контексте бизнес-требований?
    - Список всех пользователей системы
    - **Основные характеристики пользователей**
    - Цели и задачи пользователей
    - Описание типичного поведения пользователей
12. Какие из перечисленных вопросов НЕ используются при формировании бизнес-требований?

    Что система должна делать?

    Как система будет использоваться?

    Кто будет использовать систему?

    Когда система будет использоваться?
13. Какое утверждение верно в отношении бизнес-требований в сравнении с функциональными требованиями?
    - Бизнес-требования и функциональные требования описывают одно и то же.
    - Бизнес-требования являются частью функциональных требований.
    - Функциональные требования являются частью бизнес-требований.
    - **Бизнес-требования описывают, как система должна работать, а функциональные требования - что она должна делать.**
14. Какой из перечисленных документов НЕ является результатом этапа анализа потребностей в процессе написания бизнес-требований?
    - **Документ, описывающий цели и задачи проекта.**
    - Документ, описывающий потребности пользователей.
    - Документ, описывающий функциональность системы.
    - Документ, описывающий ограничения и риски проекта.
15. Что из перечисленного является примером “функциональных требований”?
    - Требования к производительности системы.
    - Требования к интерфейсам системы.
    - Требования к безопасности системы.
    - **Требования к архитектуре системы**
16. Что является первым шагом в процессе формирования бизнес-требований?

- Определение целей и задач проекта.
- **Анализ потребностей пользователей.**
- Определение ограничений проекта.
- Определение функциональных требований.

ПК-2 Способен выполнять проектирование программных систем среднего и крупного масштаба сложности
ПК-2.2 Разрабатывает концепцию программной системы

1. Что является основным результатом этапа анализа потребностей в процессе разработки концепции программной системы?
    - Список требований к системе.
    - Описание ограничений проекта.
    - Формулировка целей проекта.
    - Определение функций системы
2. Какая задача решается на этапе формирования требований в разработке концепции программной системы?
    - Выработка предложений по реализации функций системы.
    - Разработка архитектуры системы.
    - Анализ рисков проекта.
    - Оценка производительности системы.
3. Какая задача НЕ решается на этапе выработки предложений в разработке концепции программной системы?
    - Определение основных функций системы.
    - Выработка требований к системе.
    - Формирование общего видения проекта.
    - Разработка прототипа системы.
4. Что является результатом этапа оценки и выбора предложений в разработке концепции программной системы?
    - Оценка рисков и ограничений проекта.
    - Формирование списка требований к системе.
    - Создание функциональной спецификации системы.
    - Выбор оптимального решения для реализации системы.
5. Что НЕ является этапом разработки концепции программной системы?
    - Оценка и выбор предложений.
    - Разработка требований к системе.
    - Описание ограничений и рисков проекта.
    - Анализ потребностей пользователей.

4.2 Типовые контрольные вопросы и задания к экзамену (1-й семестр)

1. Основы ООП

- Дайте определение класса, типичной структуры классов
- Чем отличается процедурное программирование от объектно-ориентированного. В чем их достоинства и недостатки
- В чем заключается принцип абстракции данных
- Какие существуют отношения между классами, в чем их отличие?
- Какие бывают виды классов, в чем их отличие друг от друга?
- Какие бывают уровни доступа к членам класса, охарактеризуйте их
- Дайте определение области видимости классов и членов класса, чем область видимости отличается от области доступа
- Дайте определение абстрактного класса, опишите их отличие от обычных классов
- Что такое поле класса. В чем отличие обычных полей класса от статических полей
- Дайте определение объекта. Что это за сущность, для чего используется

- Почему не рекомендуется использовать доступ к полям Public и Protected, приведите примеры
- Дайте определение наследования, какие виды наследования бывают, чем отличаются
- Дайте определение инкапсуляции, каковы принципы ее использования
- Дайте определение полиморфизма, каковы принципы его использования
- Дайте определение интерфейса, каковы основные принципы работы с интерфейсами классов
- Дайте определение конструктора, в чем его основные особенности использования
- Дайте определение деструктора, в чем его основные особенности использования
- Дайте определение абстрактного метода, в чем его отличие от обычного, виртуального
- Дайте определение виртуального метода, в чем его отличие от обычного, абстрактного

2. Шаблоны проетирования
- Порождающие шаблоны
- Структурные шаблоны
- Поведенческие шаблоны
- Фабричный метод
- Абстрактная фабрика
- Строитель
- Фабрика прототипов
- Синглетон

4.3 Типовые вопросы к зачету по дисциплине (2-й семестр)

1. Модульные тесты
2. Потоки
3. Делегаты
4. События
5. Функциональные требования
6. Нефункциональные требования
7. Клиент-серверная архитектура
8. Компонентная архитектура
9. Объектно-ориентированная архитектура
10. Сервисная архитектура

4.4 Типовые задачи на зачет и экзамен по дисциплине

Спроектировать иерархию классов на одну из заданных тем:

- Инструменты для косьбы
- Двигатели
- Лодки
- Самолеты
- Бритвы
- Телевизоры
- Плееры
- Замки для закрывания/открывания
- Средства для приготовления еды (не посуда, а на чём готовят)
- Музыкальные инструменты
- Поезда
- Часы
- Пылесосы
- Утюги
- Миксеры/блендеры
- Печи
- Варка кофе
- Фены

- Стиральные машины

Для спроетированной иерархии реализовать один из шаблонов проектирования:

- Абстрактная фабрика
- Фабричный метод
- Строитель
- Прототип

4.5 Задание на курсовую работу

Выбрать, согласовать с преподавателем и разработать игровую программу по следующим требованиям:

Разработать игровую программу согласно заданию. Использование готовых frameworks (XNA, Unity и пр.) нельзя.

- При написании кода программы необходимо придерживаться соглашения по оформлению кода;
- Обязательно необходимо использовать шаблон проектирования MVC (Модель-Вид-Контроллер);
- Необходимо вынести общую логику работы программы, а также общие классы шаблонов MVC и пр. в отдельные библиотеки;
- Взаимодействие между классами реализовать на собственных событиях, к которым приводить события из различных реализаций программы;
- Должны быть реализованы две реализации интерфейса программы, основанные на разработанных общих классах и библиотеках:
  - WPF – полностью графический интерфейс (использование элементов управления запрещено);
  - Консольное приложение;
- В общих библиотеках/проектах не должно быть специфичного кода, относящегося к консольному или оконному приложению. Специфичная для каждой реализации логика должна быть выделена в отдельном проекте для конкретной реализации;
- Дополнительно, необходимо использовать не менее двух шаблонов проектирования;
- В визуальных формах должен отсутствовать код, связанный с логикой работы приложения (назначение форм – только показывать данные и получать данные от пользователя);
- В приложении обязательно должна использоваться многопоточность и синхронизация между потоками;
- Для одного из классов приложения должны быть разработаны полноценные модульные тесты, обеспечивающие проверку корректности его работы (не менее 20 различных тестов);
- Проекты, состоящие из одной формы без классов, служащих для обработки логики рассматриваться не будут;
- Проект программы, а также документация должны вестись в Git репозитории университета GitLab:
  - Для курсового проекта создаётся отдельный проект со структурой внутри "Решения" (Solution);
  - Разработка общих библиотек должна вестись в основной ветке проекта (master)
  - Разработка графического и консольного приложений и логики, связанной с ними должна вестись в отдельных ветках проекта (console и windows). Для каждого типа приложения ветка, делается из основной;

Результат разработки должен быть "залит" (merge) в четвёртую ветку (release), где, в итоге, должно остаться решение (Solution) с подключенными проектами как для общих библиотек, так и для двух вариантов реализации.