Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Кафедра вычислительной техники

Паспорт экзамена

по дисциплине «Технологии автоматизации процесса разработки программного обеспечения», 2 семестр

1. Методика оценки

Экзамен проводится в устной форме по билетам. Билет формируется в виде 20 тестовых вопросов из разных тем (темы – из списка экзаменационные вопросов, тестовые вопросы выбираются случайным образом из закрытого списка. На каждый тестовый вопрос дается короткий письменный ответ 1-2 абзаца. Тест оценивается 0..2 балла, суммарно 0..40.

Таким образом, проверяются результаты обучения по дисциплине, соотнесенные с установленными в программе индикаторами достижения компетенций.

На экзамене преподаватель вправе задавать студенту уточняющие и дополнительные вопросы из общего перечня (п. 4).

Форма экзаменационного билета

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Факультет АВТФ

Билет № _____
к экзамену по дисциплине «Технологии автоматизации процесса разработки программного обеспечения»

Тест 1 ----------
Тема: 19. Проектирование протокола. Стратегия взаимодействия клиента с сервером. Архитектурные аспекты проектирования протокола. Использование конечных автоматов (диаграмм состояний) в протоколах
6. Определите сущность, достоинства и недостатки стратегии: распределенная подписка клиентов на события, принимаемые сервером и создаваемые его службами.

Тест 2 ----------
Тема: 2. Поведенческие шаблоны command, iterator, mediator, snapshot, observer, state, strategy, method template, visitor. Системные шаблоны model-view-controller (MVC), session, transaction
4. Опишите логику работы паттерна observer (наблюдатель): классы, интерфейсы, структуры данных. Как в этой схеме реализуются алгоритмы действий: подписаться на получение события, отписаться от получения, передать события.

Тест 3 ----------
Тема: 16. Основные технологические решения в протоколах низких уровней и их использование в прикладном программировании сетевых приложений.
14. Что делает передающая сторона при приеме подтверждения сообщения с номером k с очередью передаваемых сообщений? Используется подтверждение приема с

номером очередного ожидаемого к приему сообщения.

Тест 4 ----------
Тема: 8. Java. Исключения, их генерация и обработка. Классы исключений. Секция finally.
7. Какие варианты обработки ошибок, кроме исключений, можно использовать в программе (2-4). Достоинства и недостатки их и исключений.

Тест 5 ----------
Тема: 3. Шаблоны параллелизма Single Threaded Execution, Two-phase Termination, Asynchronous Task, Lock Object, Read/Write Lock, Scheduler, Double Buffering, Producer-consumer
1. Опишите структуру и алгоритм работы паттерна «планировщик (sheduler)» в терминах – поток, засыпает, пробуждается, очередь, список, элемент запроса, исполняемый код, вставка/удаление из очереди (3 балла).

Тест 6 ----------
Тема: 10. Java. Потоки и их программирование. Класс Thread и интерфейс Runnable. Связь потока с классом-прародителем. Прямое управление потоками suspend,resume) и синхронизация потоков, ожидание событий (wait,notify).
9. Может ли поток самостоятельно заблокировать себя (перевести в состояние ожидания)?

Тест 7 ----------
Тема: 18. Понятие тонкого и толстого клиента. Слои: представление, контроллер, бизнес-слой, DAO. Уровни соединения клиента и сервера в типовом приложении. Прикладной протокол как элемент взаимодействия слоев клиент-серверной архитектуры. Проектирование протоколов распределенных систем.
5. Между какими слоями включено сетевое взаимодействие в web-приложении?

Тест 8 ----------
Тема: 9. Java. Программирование ввода-вывода. Классы потоков ввода-вывода. Классы - источники данных, классы представления данных. Форматы представления текста и особенности работы с ним в Java. Ввод-вывод текстовых файлов. Классы Reader,Writer. Буферизованный ввод-вывод. Понятие двоичного файла. Форматы внутреннего представления примитивных типов данных: целое со знаком и без, символ, вещественное. Двоичные потоки в Java. Объектовые потоки. Сериализация. Алгоритм сериализации системы связанных объектов.
4. Почему в двоичных потоках Data…Stream нельзя использовать методы readInt, readDouble для чтения данных из двоичных файлов, записанных не на Java?

Тест 9 ----------
Тема: 10. Java. Потоки и их программирование. Класс Thread и интерфейс Runnable. Связь потока с классом-прародителем. Прямое управление потоками suspend,resume) и синхронизация потоков, ожидание событий (wait,notify).
6. Определите термина процесс и поток применительно к ОС. Каким образом процесс и поток реализуются в Java?

Тест 10 ----------
Тема: 7. Java. Наследование в Java. Древовидная система классов при одиночном наследовании. Класс Object, его назначение. Приведение типов ссылок при работе с системой наследуемых классов. Работа с данными базового класса. Доступ к базовому классу (super). Встроенный полиморфизм методов в Java. Интерфейс (interface). Использование интерфейсов для доступа к объектам различных классов. Наследование

интерфейсов. Абстрактный базовый класс.
1. Сформулируйте основные проблемы использования и реализации множественного наследования, реализованного в C++.

Тест 11 ----------
Тема: 11. Java. Вложенные и анонимные классы. Видимость текущего объекта и объекта-родителя. Использование вложенные классов для создания взаимодействующих потоков, CallBack-вызовов и обработчиков событий. Обработка событий в Java. Классы событий, интерфейсы слушателей, объявление слушателей событий. Способы создания обработчиков событий: использование общего класса-обработчика, классов-адаптеров, анонимных классов. Функциональное программирование. Лямбда-выражения в Java.
13. Класс сетевого соединения при обращении к серверу дает множество видов ответов при единственном обращении (нормальное завершение, ошибка сети, ошибка сервера, тайм-аут). Как организовать обработку таких сообщений с помощью обратного вызова?

Тест 12 ----------
Тема: 2. Поведенческие шаблоны command, iterator, mediator, snapshot, observer, state, strategy, method template, visitor. Системные шаблоны model-view-controller (MVC), session, transaction
3. Почему итератор удобно реализовать как вложенный класс в классе структуры данных, для которой он применяется. Как сторонний код может создавать такой движок, если объект вложенного класса создается только родителем?

Тест 13 ----------
Тема: 26. RestAPI. Основные положения. Клиентская библиотека Retrofit.
6. Расшифруйте аннотацию Retrofit
@POST("/api/bug/add")
Call<String> sendBug(@Header("SessionToken") String token,@Body BugMessage bug);

Тест 14 ----------
Тема: 1. Производящие шаблоны builder, factory, prototype, singleton. Структурные шаблоны adapter, bridge, composite, decorator, facade, flyweight, proxy
4. Опишите конструкцию «адаптер обратного вызова» в терминах – интерфейс события, класс-клиент, класс-сервер, объект-адаптер, контекст клиента, опишите его работу в виде последовательности действий или коммуникационной диаграммы.

Тест 15 ----------
Тема: 11. Java. Вложенные и анонимные классы. Видимость текущего объекта и объекта-родителя. Использование вложенные классов для создания взаимодействующих потоков, CallBack-вызовов и обработчиков событий. Обработка событий в Java. Классы событий, интерфейсы слушателей, объявление слушателей событий. Способы создания обработчиков событий: использование общего класса-обработчика, классов-адаптеров, анонимных классов. Функциональное программирование. Лямбда-выражения в Java.
12. Создает ли компилятор отдельный класс для анонимного класса? Можно ли создать несколько объектов для одного анонимного класса?

Тест 16 ----------
Тема: 26. RestAPI. Основные положения. Клиентская библиотека Retrofit.
3. Почему в RESTAPI не может быть реализована полиморфная реализации методов доступа к объекту по имени сущности и ее id в БД? По «идейным» соображениям или способу реализации?

Тест 17 ----------
Тема: 8. Java. Исключения, их генерация и обработка. Классы исключений. Секция finally.
2. Поясните определение: исключение - альтернативный поток управления при возврате из цепочки вызовов в процессе обработки ошибки? Как это отображается в императивной записи кода, описывающего действия в одном потоке управления?

Тест 18 ----------
Тема: 17. Системы клиент-сервер. Клиент и сервер как программные компоненты и роли. Синхронное и асинхронное взаимодействие.
3. Определите термины «синхронный» и «асинхронный» применительно к параллельным процессам и их взаимодействиям

Тест 19 ----------
Тема: 10. Java. Потоки и их программирование. Класс Thread и интерфейс Runnable. Связь потока с классом-прародителем. Прямое управление потоками suspend,resume) и синхронизация потоков, ожидание событий (wait,notify).
10. Что означает термин «объект синхронизации» в секции synchronized и в примитивах wait/notify? Что может быть использовано в качестве такого объекта?

Тест 20 ----------
Тема: 19. Проектирование протокола. Стратегия взаимодействия клиента с сервером. Архитектурные аспекты проектирования протокола. Использование конечных автоматов (диаграмм состояний) в протоколах
11. Какие виды действий являются выходными (реакциями автомата), управляющего протоколом?

Утверждаю: зав. кафедрой ____ _____ должность, ФИО
(подпись)

2. Критерии оценки результатов обучения, соотнесенных с уровнями освоения индикаторов достижения компетенций

Ответ на экзаменационный билет засчитывается на **продвинутом** уровне, если студент проводит сравнительный комплексный анализ материала, выявляет проблемы, предлагает механизмы их решения, представляет количественные характеристики определенных процессов, приводит конкретные примеры, не допускает ошибок. Совокупность результатов обучения по дисциплине (модулю) и соотнесенных с ними индикаторов достижения компетенций не содержит пробелов. Установленные в программе компетенции сформированы в полном объеме. Оценка составляет *от 35 до 40 баллов.*

Ответ на экзаменационный билет засчитывается на **базовом** уровне, если студент при ответе на вопросы формулирует основные понятия, дает характеристику процессов, явлений, проводит анализ причин, условий, способен представить количественные и качественные характеристики процессов. Совокупность результатов обучения по дисциплине (модулю) и соотнесенных с ними индикаторов достижения компетенций содержит несущественные пробелы. Установленные в программе компетенции сформированы на базовом уровне. Оценка составляет *от 34 до 27 баллов.*

Ответ на экзаменационный билет засчитывается на **пороговом** уровне, если студент при ответе на вопросы дает определение основных понятий, может показать причинно-следственные связи явлений. Совокупность результатов обучения по дисциплине (модулю) и соотнесенных с ними индикаторов достижения компетенций содержит

пробелы. Установленные в программе компетенции сформированы на пороговом уровне. Оценка составляет *от 26 до 20 баллов.*

Ответ на экзаменационный билет считается **неудовлетворительным** (ниже порогового уровня), если студент при ответе на вопросы не дает определений основных понятий, не способен показать причинно-следственные связи явлений. Совокупность результатов обучения по дисциплине и соотнесенных с ними индикаторов достижения компетенций содержит существенные пробелы. Установленные в программе компетенции не сформированы. Оценка составляет *менее 20 баллов.*

3. Шкала оценки

Экзамен считается сданным, если сумма баллов по всем заданиям билета составляет от 20 до 40 баллов включительно. Сумма менее 20 баллов признается неудовлетворительным результатом промежуточной аттестации по дисциплине.

В общей оценке по дисциплине экзаменационные баллы учитываются в соответствии с правилами балльно-рейтинговой системы, установленными в НГТУ.

4. **Вопросы к** экзамену **по дисциплине «**Технологии автоматизации процесса разработки программного обеспечения**»**

1. Java. Базовые типы данных. Класс String, его особенности и методы. Классы – обертки для базовых типов данных (int и Integer), их назначение. Базовые типы данных и ссылочные типы. Особенности работы со ссылочными типами, динамическое распределение памяти, «сбор мусора». Операции над ссылками.
2. Java. Наследование и приведение типов. Класс Object. Расширение и сужение ссылочного типа при переходе от класса к классу. Операция instanceof. Встроенный полиморфизм методов. Java. Объектная модель JVM. Классы Object и Class, их методы. Организация JVM. Система мета-объектов – описателей классов. Динамическая загрузка классов в JVM.
3. Java. Статические элементы классов. Их использование для задания констант и функций. Метод main. Модульная структура Java-программы. Пакет, класс, метод. Полные и сокращенные имена классов и методов. Объявление и подключение (импорт) пакетов. Области видимости: проект, пакет, класс, метод, производный класс. Модификаторы доступа (public, private, protected) и модификаторы изменения (static, final). Инициализаторы и конструкторы. Многообразие вариантов инициализации объектов. Явный вызов конструкторов. Массивы в Java. Массив как ссылочный тип и как объект. Реализация объекта-массива как динамического массив указателей (ссылок). Размерность массива, инициализация массивов. Многомерные массивы.
4. Java. Наследование в Java. Древовидная система классов при одиночном наследовании. Класс Object, его назначение. Приведение типов ссылок при работе с системой наследуемых классов. Работа с данными базового класса. Доступ к базовому классу (super). Встроенный полиморфизм методов в Java. Интерфейс (interface). Использование интерфейсов для доступа к объектам различных классов. Наследование интерфейсов. Абстрактный базовый класс.
5. Java. Исключения, их генерация и обработка. Классы исключений. Секция finally.
6. Java. Программирование ввода-вывода. Классы потоков ввода-вывода. Классы -

источники данных, классы представления данных. Форматы представления текста и особенности работы с ним в Java. Ввод-вывод текстовых файлов. Классы Reader,Writer. Буферизованный ввод-вывод. Понятие двоичного файла. Форматы внутреннего представления примитивных типов данных: целое со знаком и без, символ, вещественное. Двоичные потоки в Java. Объектовые потоки. Сериализация. Алгоритм сериализации системы связанных объектов.

7. Java. Потоки и их программирование. Класс Thread и интерфейс Runnable. Связь потока с классом-прародителем. Прямое управление потоками suspend,resume) и синхронизация потоков, ожидание событий (wait,notify).
8. Java. Вложенные и анонимные классы. Видимость текущего объекта и объекта-родителя. Использование вложенные классов для создания взаимодействующих потоков, CallBack-вызовов и обработчиков событий. Обработка событий в Java. Классы событий, интерфейсы слушателей, объявление слушателей событий. Способы создания обработчиков событий: использование общего класса-обработчика, классов-адаптеров, анонимных классов. Функциональное программирование. Лямбда-выражения в Java.
9. Java. Рефлексия. Сериализация. JNI
10. Scala. Основные идеи. Архитектура. Синтаксис. Классы и объекты. Прямое и обратное взаимодействие кода Scala – Java. Шаблоны (generic) как базовый элемент Scala. Массивы как объекты. Конструкторы. Синтаксис. Передача параметров при наследовании.
11. Scala. Внутренняя мета-модель языка. Функции как классы, константы как объекты, операции как методы (переопределение операций). Объектная модель языка. Классы Any, AnyVal, AnyRef. Приведение типов. Объектная модель языка. Trait-ы. Их отличие от интерфейсов.
12. Scala. Функции как параметры. Анонимные функции. Scala. Контейнеры типов (монады). Контейнеры Option для null-значения. Контроль наследования типов. Вариантность. Ковариантность и контрвариантность.
13. Scala. Структуры данных: Array, List, Map, кортеж. Операции над ними. Функциональные комбинаторы. Императивный и функциональный стиль программирования в Scala. Вложенные функции.
14. Kotlin. Особенности синтаксиса. «Инвертированная» конкатенация строки с переменными и выражениями, префиксный синтаксис. Статическое выведение типа.
15. Kotlin. Иерархия классов. Классы Any и Nothing. Синтаксический контроль null-pointer.
16. Kotlin. Переопределение операций.
17. Kotlin. Циклы. Циклы для классов с итераторами.
18. Kotlin. Объекты как статические компоненты класса и синглетоны. Анонимные, вспомогательные и декларированные объекты.
19. Kotlin. Вложенные, внутреннние и анонимные классы. Функциональные интерфейсы, анонимные функции (лямбда-выражения).
20. Kotlin. Функциональный тип. Переменная функционального типа: определение, ФП, результат. Метод класса – функциональный тип в контексте класса.
21. Kotlin. Функции области видимости.

22. Kotlin. Параллелизм. Классическое определение сопрограмм. Модель сопрограмм на потоке управления и на взаимно блокирующихся потоках. Реализация сопрограмм на уровне компиляции в байт-код: класс сопрограммы, пул потоков, планировщик. «Синтаксические преимущества» использования сопрограмм в сравнении с обычными потоками и CallBack-завершениями. Реализация delay и примитивов асинхронного запуска и ожидания завершения.
23. Сравнительный анализ Java, Scala,Kotlin. Особенности синтаксиса. Префиксная и контекстная формы определения элементов синтаксиса.
24. Сравнительный анализ Java, Scala,Kotlin. Особенности иерархии классов и компиляции в байт-код.
25. Сравнительный анализ C++, Java, Scala,Kotlin. Особенности реализации шаблонов.
26. Сравнительный анализ C++, Java, Scala,Kotlin. Особенности реализации массивов.
27. Сравнительный анализ C++, Java, Scala,Kotlin. Особенности определения свойств класса, конструирования (в т.ч. при наследовании)
28. Сравнительный анализ C++, Java, Scala,Kotlin. Особенности переопределения операций
29. Сравнительный анализ C, C++, Java, Scala,Kotlin. Особенности использования функций. Функция как тип данных
30. Функциональный и императивный стили программирования. Функция как объект, «чистая» функция (метод).
31. Архитектура Android. Уровни. Их реализация, программирование, Java,C++. Реализация на основе Linux. Особенности виртуальной машины Dalvik. Файловая система: разделы внутренней памяти, логическая структура ФС. Особенности управления процессами и памятью. Особенности внутреннего программирования. Activity. События, жизненный цикл.
32. Android. Архитектура ОС Android. Ядро. Библиотеки. Среда исполнения (Dalvik-машина). Уровень каркаса приложений. Виды приложений (действия, сервисы, источники данных, приемники). Особенности компиляции приложений: файлы dx и apk. Система разработки для Android. Структура Android- приложения. Файл AndroidManifest.xml. Описание компонент (действий, служб) и разрешений
33. Android. Класс Activity. Классы элементов изображения и менеджеров компоновки View, Button, TextView, EditText, LinearLayout, TableLayout, TableRow. Статическое описание разметки в XML-файлах, доступ по идентификаторам ресурсов, назначение обработчиков событий. Динамическая (программная разметка). Диалоги с динамической (программной) и статической XML-разметкой
34. Android. Особенности выполнения приложений, связанные с ресурсными ограничениями. Класс Activity, его жизненный цикл, состояния и события. Запуск Activity, передача данных и возврат результата от Activity. Сохранение данных при выгрузке. Намерения, межпрограммные коммуникации. Широковещательные сообщения между Activity и приложениями.
35. Понятие программной ошибки. Философия ошибок. Классификационные характеристики ошибок. Отладка, инспекция и тестирование как этапы поиска ошибок Ошибки вычислений и преобразований. Ошибки структурирования кода. Ошибки форматов входных данных. Ошибки форматов внутренних данных и соглашений по данным. Ошибки сборки, конфигурирования, размещения. Ошибки

использования ресурсов. Ошибки, связанные с ограничениями по ресурсам. Ошибки реактивности и производительности. Ошибки параллелизма и синхронизации. Ошибки распределенных систем и протоколов. Ошибки пользовательского интерфейса.

36. Структурное тестирование. Тестирование операторов, условий (решений) и путей. Комбинационное тестирование. Пример. Функциональное тестирование. Классы эквивалентности по входным данным. Проектирование тестового покрытия. Классы эквивалентности по граничным условиям. Примеры.

37. Отладка и инспекция программного кода. Сущность отладки. Приемы отладки. Инспекция как неформальный анализ программного кода. Логический и временной анализ. Методы инспектирования. TDD – разработка, управляемая тестированием.