Диаграмма состояний

1. Светофор *TrafficLights* после создания переходит в состояние выключен *Offline*. При включении *On*, светофор переходит во вложенное состояние «зеленый» *Green* состояния включен *Online*. По истечении 50 секунд, светофор переходит в состояние «желтый» *Yellow* в *Online*. Затем, через 3 секунды – в состояние «красный» *Red*. По истечении 50 секунд светофор возвращается в состояние «зеленый».

а. Добавьте возможность выключить *Off* включенный светофор.

б. Доработайте модель, укажите, что интервалы *t* между переключениями сигналов светофора настраиваются вызовом операций *setGreen, setRed* и *setYellow* в выключенном состоянии.

в. Измените порядок включения светофора, используя сторожевые условия, укажите по аналогии с предыдущим пунктом, что начальный сигнал светофора при включении *initialGreen, initialYellow* или *initialRed* настраивается *setInitial* в выключенном состоянии.

2. После создания статья *Paper* является черновиком *Draft*. После отправки *sent* статья рассматривается программным комитетом конференции *OnReview*, при этом выполняется рецензирование статьи *reviewPaper*. При согласии комитета *approved*, статья принимается на конференцию *Accepted*. Если комитет не принял статью *declined*, статья становится черновиком *Draft*.

а. Уточните изменения состояний статьи при рецензировании. После получения статья проверяется *Checking*. Если обнаружены недостатки, статья направляется на доработку *correct* в состояние *OnCorrection*. Если недостатков нет, рецензирование статьи завершается. При повторной отправке статьи *sent* она проверяется повторно *Checked*.

б. Укажите, что если статья не была отправлена с исправлениями недостатков в течение 10 дней, рецензирование прекращается и статья становится черновиком *Draft*.

3. Простейшие цифровые часы состоят из дисплея и двух кнопок A и B. Часы могут работать в двух режимах: отображения и настройки. В режиме отображения часы показывают часы и минуты, между которыми мигает символ двоеточия.

Режим настройки состоит из двух подрежимов: настройка часов и настройка минут. Кнопка A позволяет выбрать режим. Каждый раз при ее нажатии происходит переход к очередному режиму в последовательности: отображение, установка часов, установка минут, отображение и т. д. Кнопка B позволяет увеличивать значение часов или минут на единицу при каждом нажатии в одном из режимов установки. Чтобы кнопка смогла породить новое событие, ее необходимо отпустить. Нарисуйте диаграмму состояний часов.

4. Независимая система управления всего устройства определяет, когда двигатель должен быть включен, и непрерывно подает сигнал ВКЛ на управляющий вход двигателя.

Когда на вход подается сигнал ВКЛ, система управления двигателя должна запустить двигатель и поддерживать его работу. Двигатель запускается подачей напряжения на пусковую и рабочую обмотки. Датчик, называемый стартовым реле, определяет момент запуска двигателя, после чего отключает пусковую обмотку. Напряжение остается только на рабочей обмотке. Когда сигнал ВКЛ пропадает, обе обмотки отключаются.

Электродвигатели, применяемые в бытовой технике, могут перегреваться из-за чрезмерной нагрузки или невозможности запуска. Для защиты от перегрева в систему управления двигателем часто добавляется датчик превышения температуры. Если

двигатель нагревается слишком сильно, система управления снимает напряжение с обеих обмоток и игнорирует сигнал ВКЛ до тех пор, пока двигатель не остынет и не будет нажата клавиша сброса.

Добавьте на диаграмму следующие элементы. Деятельность: подать напряжение на пусковую обмотку, подать напряжение на рабочую обмотку. События: двигатель перегрелся, подан сигнал ВКЛ, снят сигнал ВКЛ, двигатель работает, сброс. Условие: двигатель не перегрет.

5. При выполнении проекта *Run* руководитель проекта (РП) осуществляет отслеживание текущего статуса *Monitor*. Если выявляется проблема *problem*, то РП устраняет ее *ManageProblem*, и возвращается *succeed* к отслеживанию. Если РП не удается решить проблему *fail*, то происходит завершение проекта *Shutdown* с выполнением остановки *stop*, по завершении которой поведение руководителя проекта завершается.

а. Добавьте этап работы РП по подготовке *Initiate* проекта. На этом этапе при необходимости *estim* производится оценка *preparePlan* для проектов с бюджетом *budget* более 100K. Для остальных проектов готовится *sketch* набросок. По утверждении *approve* проекта РП переходит к управлению им *Run*.

б. Покажите на диаграмме, что при управлении проектом РП взаимодействует с другими *Communicate*. Если поступает запрос на собрание m*eetingRequest*, то РП его назначает *schedule*. По завершении *done* собрания *RunMeeting*, РП возвращается к взаимодействию.

в. Добавьте возможность РП приостановить *suspend* работы в любой момент. Если работы приостановлены более года, то РП завершает проект *Shutdown*. При возобновлении *resume*, выполнение проекта продолжается с того момента, когда он был приостановлен.

г. (*) Разделите роли руководителя проекта и старшего руководителя проекта. Старший РП так же в любой момент управления проектом принимает запросы на изменения *AcceptCR*, при получении *accept* которого проводит оценку *Estimate*, отвергает *reject* или принимает *approve* с возвращением к приему. *Указание*. Добавьте дочерний класс и используйте расширение схем состояний.

6. Согласно спецификации платформы Java в части управления памятью, при создании объект находится в состоянии *Created*, при входе в которое выделяется память *alloc*, и вызываются инициализаторы и конструкторы *init*. Когда локальной переменной метода присваивается *assigned* объект, он становится используемым *InUse*. Если переменная вышла из области достижимости *outOfScope*, объект становится недоступным *Unreachable*. При обнаружении сборщиком мусора, если в классе объекта определен метод *finalize ()*, и он не был вызван, то объект помещается в очередь на сборку в состоянии *Collected*, иначе объект становится очищенным *Finalized*. После того, как память объекта возвращена *deallocated*, жизненный цикл объекта завершен.

а. Укажите, что если переменная используемого объекта выходит из блока, в котором она была определена, и других ссылок на объект нет, то объект становится невидимым *Invisible*. После завершения выполнения метода, в котором определена переменная, объект становится недоступным.

б. Покажите, что в очереди на сборку выполняется деятельность *waitFinalize*, которая завершается после завершения выполнения метода *finalize ()*. По ее завершении, если объект воскрешен *resurrected*, то есть, обнаружены ссылки на объект, то объект

становится используемым, иначе переходит в состояние *Finalized*.

в. Используя псевдосостояние выбора явно, покажите, что если в классе объекта не определен метод *finalize ()*, то недоступный объект при обнаружении сразу переходит в состояние *Finalized*.

г. Используя композитные состояния, покажите, что объект может воскреснуть, пока он невидим, недоступен, ожидает сборки или очищен.

7. Самолет *Aircraft* изначально находится *InHangar* в ангаре. При выходе на рейс *flight* самолет переходит на посадку *Boarding*. В полете *InAir* самолет выполняет деятельность по управлению полетом *flightControl*. В начале полета самолет убирает шасси *pullGearUp*, и выпускает в конце *pullGearDown*.

а. Устраните недостаток в модели – покажите, что при получении разрешения на взлет самолет переходит в состояние взлета *TakeOff* и после отрыва – в состояние полета. При получении разрешения на посадку *landingPermit* самолет переходит в *Landing* и, по прибытии, переходит на посадку.

б. В соответствии с требованиями аэропорта, самолет должен отправлять сигнал об освобождении полосы *freeRunway* после взлета и перед прибытием, и поддерживать постоянный радиоконтакт *radioComm* при нахождении на полосе.

в. Уточните состояние полета, укажите, что в долгом полете через час после взлета пассажирам предлагается обед *Dinner*, который длится один час.

г. Отразите в модели, что в случае возникновения неисправности в полете *malfunction,* самолет будет поврежден *Damaged*. Покажите, что самолет может совершить посадку только в исправном состоянии *Normal*.

д. Что произойдет, если самолет получит разрешение на посадку во время обеда? Ответ поясните.

8. Контроллер мультимедиа-проигрывателя *PlayerController* изначально находится в нерабочем состоянии *NotInitialized*. При запуске проигрывателя происходит переход в состояние загрузки *Loading*, при входе в которое инициализируются управляющий компонент *initEngine* и интерфейс проигрывателя *initUI*. В данном состоянии загружаются доступные модули *loadPlugins*. По завершению загрузки контроллер переходит в состояние готовности *Ready*.

а. Основное назначение проигрывателя – воспроизводить аудио и видео. Добавьте в состоянии готовности режимы воспроизведения, остановки и паузы, а также события, инициирующие переходы между этими режимами.

б. Хорошие мультимедиа-проигрыватели параллельно воспроизведению автоматически обновляют медиатеку: осуществляют поиск нового контента *Searching* и загружают информацию о нем из сети *Downloading*. Добавьте в модель данную функциональность.

9. Моделируется мобильное приложение – электронный словарь, имеющее несколько окон. Схему перехода между окнами будем моделировать с помощью диаграммы схем состояний.

а. Добавьте на диаграмму состояния *Список слов*, *Перевод*, *Словоформы слова*. Переход между *Список слов* и *Перевод* по событию *Выбрано слово*, обратный переход по событию *Назад*. Переход между *Перевод* и *Словоформы слова* по событию *Словоформы*, обратный переход по событию *Назад*.

б. В некоторых карточках перевода есть ссылки на другие карточки. При переходе по ссылке мы попадаем в то же состояние *Перевод*, но для другого слова. Добавьте переход в себя из состояния *Перевод*, активируемый событием *Переход по ссылке*.

в. Возврат после перехода по ссылке по событию *Назад* должен возвращать пользователя в предыдущую карточку. Для этого добавим к переходу по ссылке действие *Добавить в стек*. Добавим переход в себя из состояния *Перевод* по событию *Назад* со сторожевым условием *[стек не пуст]* и действием *Убрать из стека*. В переход из состояния *Перевод* в состояние *Список слов* по событию назад нужно добавить сторожевое условие *[else]*.

г. Перед запуском приложения нужно загрузить словари. На время загрузки словарей пользователю будет показан экран приветствия. Добавим ортогональное состояние *Инициализация*. В нем выделим два региона. В первом регионе добавим начальное и конечное псевдосостояния. Между ними добавим состояние *Загрузка* словарей с выполняемой при нахождении в состоянии деятельностью *Загрузить словари*. Ожидается переход в конечное состояние по завершении деятельности. Во втором регионе мы добавим переход из начального состояния в конечное по событию времени *after (3 s)*. Это необходимо, чтобы при быстрой загрузке пользователь успел прочитать содержимое экрана приветствия. Из ортогонального состояния *Инициализация* переход по завершению в *Список слов*.

д. Уточним поведение в окне списка слов. *Список слов* становится составным состоянием. Дальнейшие состояния являются вложенными в *Список слов*. Из начального псевдосостояния происходит переход в состояние *История запросов*. По событию *Ввод* из *Истории запросов* происходит переход в псевдосостояние выбора. Далее при условии *поле ввода пустое* происходит переход (возврат) в состояние *История запросов*. Иначе происходит переход во вложенное составное состояние *Поиск слова*. По событию *Ввод* происходит переход из *Поиск слова* в описанное выше псевдосостояние выбора. В составном состоянии *Поиск слова* непосредственно процедура поиска откладывается с помощью события времени *after (1 s)*. После этого события происходит переход из начального псевдосостояния в состояние *Поиск в словаре*. По завершению происходит переход из *Поиск в словаре* в *Отображение списка*.

10. При вызове операции пассивного открытия *listen* модуль протокола TCP переходит из *Closed* в *Listen*. При получении сообщения *SYN* будет выполнен переход в *SYN_Rcvd* с отправкой *SYN+ACK*, и далее в *Established* при получении *ACK*. При вызове операции *close* модуль переходит в *ActiveClose*, отправляя *FIN*, и через 2 секунды возвращается в *Closed*. Если же в *Established* было получено сообщение *FIN*, то модуль переходит в *PassiveClose* и по получении *ACK* переходит в *Closed*.

а. Покажите поведение модуля TCP на схеме состояний. Перечислите состояния, переходы, укажите для переходов триггеры, виды событий (получение сигнала, вызов операции и др.) и эффекты при переходе.

б. Добавьте активное открытие соединения *ActiveOpen* через отправку *SYN* при вызове операции *connect* в *Closed*, и переходе в *Established* по получении *SYN+ACK*.

в. Уточните, что в *PassiveClose* модуль до получения *ACK* ожидает однократного вызова *close*, отправляя *FIN*. При том же вызове *close*, модуль переходит в *ActiveClose* из *SYN_Rcvd* с отправкой *FIN*, в *Closed* из *ActiveOpen*.