**Техническое задание на разработку программного обеспечения.**

**Worker manager**

**Задача:**

Разработать систему, позволяющую снизить нагрузку при выполнении громоздких синхронных операций для приложений написанных с использованием технологии Adobe AIR. Под громоздкими синхронными операциями подразумеваются такие операции как разбор (parsing) больших документов, сложные математические вычисления, обработка большого количества данных, а также другие операции, занимающие неприемлимое для основного потока процессорное время, сказывающиеся на производительности отрисовки кадров.

**Решение:**

Решение предполагает написание системы базирующейся на технологии AS3 Workers. При этом внешнее использование данной системы предполагает, что стандартная в AS3 модель управления потоками (далее Воркерами) должна быть инкапсулирована внутрь. Это значит, что публичный API данной системы должен позволять управлять воркерами не прибегая к использованию таких манипуляций как работа с Worker.workerDomain, создание каналов связи, подписка на стандартные события MessageChannel и т. д., все это должно выполняться внутри данной системы.

В рамках решения задачи необходимо разработать Worker manager (далее Менеджер), представляющий собой реализацию некого контейнера, позволяющего управлять воркерами через единый интерфейс.

Менеджер должен быть реализован на основе шаблона «Одиночка» (Singleton) и предоставлять доступ к управлению воркерами. Доступ к воркерам должен основываться на вызове публичных методов Менеджера и событийной модели в рамках спецификации AS3 (addEventListener, removeEventListener, …)

Данный интерфейс должен содержать методы для запуска, остановки, мониторинга состояния и межпоточного общения между основным потоком и потоком выполнения воркера.

С точки зрения внешнего использования Менеджер должен позволять осуществлять следующие операции:

- Добавить воркер (addWorker(worker:IWorkerUnit):void )

- Удалить воркер (removeWorker(worker:IWorkerUnit):void )

- Запросить количество добавленных воркеров ( get workerCount():uint )

- Запросить общее текущее состояние Менеджера (информативная реализация toString)

- Запросить конкретный воркер по идентификатору ( getWorkerByID(id:uint):IWorkerUnit )

- Запросить конкретный воркер по индексу ( getWorker(uint:index):IWorkerUnit )

- … другие методы доступа, реализация которых будет полезной в контексте данного решения

Воркеры, с которыми работает Менеджер должны быть представлены в виде объектов реализующих интерфейс IWorkerUnit.

Классы, реализующие этот интерфейс внутри делегируют загрузку байтовых данных, т. е. непосредственно загрузка байтовых данных воркера осуществляется внутри этих классов.

Чтобы инкапсулировать загрузку байтовых данных, и всю работу со стандартным AS3 Workers API, необходимо реализовать класс WorkerUnit, который будет делегировать эти операции внутри себя.

Соответственно, все воркеры, использование которых предполагается в Менеджере, будут наследниками этого класса.

Пример наследования:

 package com.addreality.system.async {

/\*\*

 \* Пример реализаций одного из воркеров, который может быть использован в WorkerManager-e.

 \*/

public class PSDParserWorkerUnit extends WorkerUnit {

 // Embedding файла SWF

 [Embed(source="../../../../../../bin/psd\_parser\_worker.swf", mimeType="application/octet-stream")]

 private var WorkerByteArray:Class;

 public PSDParserWorkerUnit() {

 // Суперкалссу (WorkerUnit) передаем байтовые данные SWF файла

 // с которыми он у себя внутри работает (внутренние подписки на события,

 // каналы связи, и т.д.)

 super(new WorkerByteArray());

 }

 }

 }

И далее, экземпляр данного класса может быть помещен в Менеджер, например следующим образом:

WorkerManager.instance.addWorker(new PSDParserWorkerUnit());

**Здесь описан не весь API системы, а только общая идея, поэтому собственные идеи Исполнителя касательно проектирования весьма желательны.**

Внутренняя реализация механизма системы ложится на плечи исполнителя, но код должен быть масштабируемым и понятным.