**Общая информация**

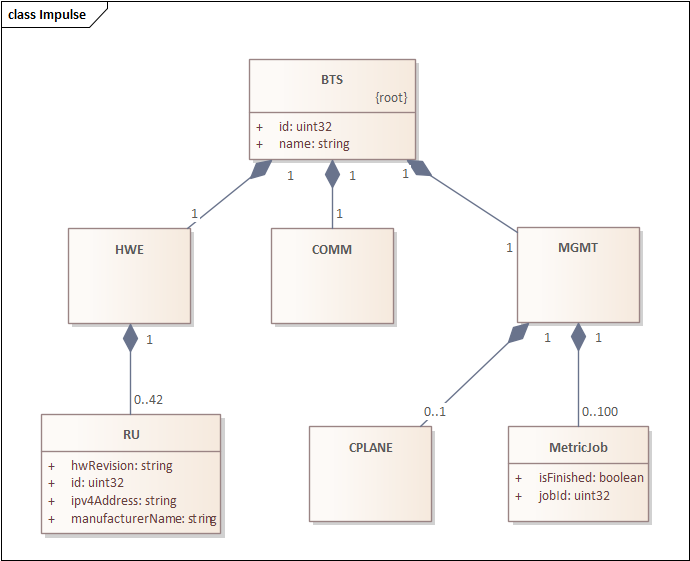
*Базовая станция* – сложный аппаратно-программная система по приему и передаче радиосигналов, способная работать со множеством мобильных устройств одновременно. Аппаратная часть состоит в том числе из антенны и одного или нескольких радиомодулей, связанных между собой физически. *Программная часть* состоит из большого количества сервисов, работающих вместе на одном «железе», при этом каждый из них отвечает за конкретный функционал. Координирующим узлом в программной части выступает *система управления базовой станцией*.

Важной частью системы управления базовой станции является *информационная модель*, которая представляет собой формальную UML-диаграмму с описанием иерархической структуры *классов и их атрибутов*. Данная модель разрабатывается в специализированных программах, которые предоставляют удобный интерфейс для одновременной совместной работы над моделями, хранение их в БД, разграничение доступа. В процессе эволюции системы в модель регулярно вносятся изменения (появляются новые классы, изменяются атрибуты и т.д.).

Модель в виде рисунка понятна и привычна для человека, но для компьютерной обработки удобнее другой формат, более приближенный к машинному, поэтому модель может быть переведена в *XML-файл*, однозначно соответствующий UML-диаграмме.

Полученный XML проверяется на непротиворечивость и отсутствие ошибок (при разработке всегда может сыграть человеческий фактор) – этот процесс мы называем *валидацией*. Если ошибок нет, запускается *генерация* определенных частей исходного кода – *артефактов* – по сути, интерфейсов для программ всех компонент базовой станции (к ним относятся файлы конфигурации, YANG-файлы, исходники на С++ и Go, Proto-файлы и т.д.). Одновременный переход всех компонент на новую версию артефактов позволяет избежать ошибок несовместимости версий и является одной из первоочередных задач системы управления базовой станцией.

**Информационная модель**

Для тестового задания предлагается некоторая упрощенная модель, из которой убраны лишние связи и атрибуты, но тем не менее она использует те же принципы, что и основная ее версия:  


XML-файл модели состоит из двух логических частей – описание классов и описание связей между ними. Каждый класс содержит описание атрибутов (если они есть), каждый атрибут имеет тип. Связи между классами – агрегации. Для каждой из них указано два имени – target (класс, включающий в себя другой класс) и source (класс, включенный в другой класс). Также в отношениях агрегации указаны мощности (multiplicity) – может быть как одно число, так и диапазон [min..max].

Файл: **impulse\_test\_input.xml**

Json-файл **config.json** содержит в себе конфигурационные данные модели. Для упрощения формат был сокращен до одного уровня. **patched\_config.json —** файл конфигурации после применения к нему некоторых изменений.

Файл: **config.json, patched\_config.json**

**Задание**

Требуется написать программу на Python, которая на основе входных файлов (**impulse\_test\_input.xml, config.json, patched\_config.json**) генерирует следующие виды выходных файлов (артефактов):

1. Файл **config.xml** представляет собой пример внутренней конфигурации базовой станции, соответствующий представленной модели.
2. Файл **meta.json** содержит мета-информацию о классах и их атрибутах. Фронтенд использует данный файл для корректного отображения дерева объектов в интерфейсе пользователя (UI).
3. Файл **delta.json** содержит в себе дельту, отражающую разницу между файлами **config.json** и **patched\_config.json.** Имеет в себе поля   
   **additions —** новые поля, которые есть в **patched\_config.json,** но отсутствуют в **config.json , deletions —** удаленные параметры. Имеются в **config.json,** но отсутствуют в **patched\_config.json  
   updates —** поля, значения которых были изменены.  
   Вид **delta.json** с примерами для каждого из полей продемонстрирован в приложении 5.
4. Файл **res\_patched\_config.json** содержит в себе результат применения полученной дельты (**delta.json)** к файлу **config.json.** Порядок параметров в файлах **patched\_config.json** и **res\_patched\_config.json** может отличаться.

Ожидаемую структуру выходных файлов можно посмотреть в приложенных файлах-примерах.

**Приложение 1. Входной файл impulse\_test\_input.xml**

<?xml version="1.0" ?>

<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3" timestamp="2024-05-11 12:34:56">

    <Class name="BTS" isRoot="true" documentation="Base Transmitter Station. This is the only root class">

        <Attribute name="id" type="uint32" />

        <Attribute name="name" type="string" />

    </Class>

    <Class name="MGMT" isRoot="false" documentation="Management related">

    </Class>

    <Class name="COMM" isRoot="false" documentation="Communication services">

    </Class>

    <Class name="MetricJob" isRoot="false" documentation="Perfomance metric job">

        <Attribute name="isFinished" type="boolean" />

        <Attribute name="jobId" type="uint32" />

    </Class>

    <Class name="CPLANE" isRoot="false" documentation="Perfomance metric job">

    </Class>

    <Class name="RU" isRoot="false" documentation="Radio Unit hardware element">

        <Attribute name="hwRevision" type="string" />

        <Attribute name="id" type="uint32" />

        <Attribute name="ipv4Address" type="string" />

        <Attribute name="manufacturerName" type="string" />

    </Class>

    <Class name="HWE" isRoot="false" documentation="Hardware equipment">

    </Class>

    <Aggregation source="MGMT" target="BTS" sourceMultiplicity="1" targetMultiplicity="1" />

    <Aggregation source="HWE" target="BTS" sourceMultiplicity="1" targetMultiplicity="1" />

    <Aggregation source="COMM" target="BTS" sourceMultiplicity="1" targetMultiplicity="1" />

    <Aggregation source="MetricJob" target="MGMT" sourceMultiplicity="0..100" targetMultiplicity="1" />

    <Aggregation source="CPLANE" target="MGMT" sourceMultiplicity="0..1" targetMultiplicity="1" />

    <Aggregation source="RU" target="HWE" sourceMultiplicity="0..42" targetMultiplicity="1" />

</XMI>

**Приложение 2. Пример выходного файла config.xml**

<BTS>

    <id>uint32</id>

    <name>string</name>

    <MGMT>

        <MetricJob>

            <isFinished>boolean</isFinished>

            <jobId>uint32</jobId>

        </MetricJob>

        <CPLANE>

        </CPLANE>

    </MGMT>

    <HWE>

        <RU>

            <hwRevision>string</hwRevision>

            <id>uint32</id>

            <ipv4Address>string</ipv4Address>

            <manufacturerName>string</manufacturerName>

        </RU>

    </HWE>

    <COMM>

    </COMM>

</BTS>

**Приложение 3. Пример выходного файла meta.json**

[

    {

        "class": "MetricJob",

        "documentation": "Perfomance metric job",

        "isRoot": false,

        "max": "100",

        "min": "0",

        "parameters": [

            {

                "name": "isFinished",

                "type": "boolean"

            },

            {

                "name": "jobId",

                "type": "uint32"

            }

        ]

    },

    {

        "class": "CPLANE",

        "documentation": "Perfomance metric job",

        "isRoot": false,

        "max": "1",

        "min": "0",

        "parameters": []

    },

    {

        "class": "MGMT",

        "documentation": "Management related",

        "isRoot": false,

        "max": "1",

        "min": "1",

        "parameters": [

            {

                "name": "MetricJob",

                "type": "class"

            },

            {

                "name": "CPLANE",

                "type": "class"

            }

        ]

    },

    {

        "class": "RU",

        "documentation": "Radio Unit hardware element",

        "isRoot": false,

        "max": "42",

        "min": "0",

        "parameters": [

            {

                "name": "hwRevision",

                "type": "string"

            },

            {

                "name": "id",

                "type": "uint32"

            },

            {

                "name": "ipv4Address",

                "type": "string"

            },

            {

                "name": "manufacturerName",

                "type": "string"

            }

        ]

    },

    {

        "class": "HWE",

        "documentation": "Hardware equipment",

        "isRoot": false,

        "max": "1",

        "min": "1",

        "parameters": [

            {

                "name": "RU",

                "type": "class"

            }

        ]

    },

    {

        "class": "COMM",

        "documentation": "Communication services",

        "isRoot": false,

        "max": "1",

        "min": "1",

        "parameters": []

    },

    {

        "class": "BTS",

        "documentation": "Base Transmitter Station. This is the only root class",

        "isRoot": true,

        "parameters": [

            {

                "name": "id",

                "type": "uint32"

            },

            {

                "name": "name",

                "type": "string"

            },

            {

                "name": "MGMT",

                "type": "class"

            },

            {

                "name": "HWE",

                "type": "class"

            },

            {

                "name": "COMM",

                "type": "class"

            }

        ]

    }

]

**Приложение 4 (A). Пример входного файла config.json**

{

"param0": "89",

"param1": "475",

"param2": "617",

"param3": "927",

"param4": "665",

"param5": "2",

"param6": "5",

"param7": "152",

"param8": "153",

"param9": "431"

}

**Приложение 4 (Б). Пример входного файла patched\_config.json**

{

"param1": "1672",

"param2": "1740",

"param4": "665",

"param5": "2",

"param8": "1058",

"param9": "1722",

"added\_param0": "2471",

"added\_param1": "2946",

"added\_param2": "2701",

"added\_param3": "2733"

}

**Приложение 5. Пример выходного файла delta.json**

{

"additions": [

{

"key": "added\_param0",

"value": "2471"

},

{

"key": "added\_param1",

"value": "2946"

},

{

"key": "added\_param2",

"value": "2701"

},

{

"key": "added\_param3",

"value": "2733"

}

],

"deletions": [

"param0",

"param3",

"param6",

"param7"

],

"updates": [

{

"key": "param1",

"from": "475",

"to": "1672"

},

{

"key": "param2",

"from": "617",

"to": "1740"

},

{

"key": "param8",

"from": "153",

"to": "1058"

},

{

"key": "param9",

"from": "431",

"to": "1722"

}

]

}

**Приложение 6. Пример выходного файла res\_patched\_config.json**

{

"param1": "1672",

"param2": "1740",

"param4": "665",

"param5": "2",

"param8": "1058",

"param9": "1722",

"added\_param0": "2471",

"added\_param1": "2946",

"added\_param2": "2701",

"added\_param3": "2733"

}