|  |
| --- |
| EPAM Systems, RD Dep. |
| Практическое задание  JAVA.SE.06 Generic and Collections |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| <1.0> | Первая версия | Игорь Блинов | <04.08.2011> |  |  |
| <2.0> | Вторая версия: задания изменены согласно обновленному содержанию модуля | Ольга Смолякова | <12.11.2014> |  |  |

##### **Задание 1. Классы коллекций**

Изучите классы реализации коллекций и заполните следующую таблицу

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ordering | Random Access | Key-Value Pairs | Allows Duplicates | Allows Null Values | Thread Safe | Blocking Operations |
| ArrayList | Yes | Yes | No | Yes | Yes | No | No |
| LinkedList | Yes | No | No | Yes | Yes | No | No |
| Vector | Yes | Yes | No | Yes | Yes | Yes | No |
| Stack | Yes | No | No | Yes | Yes | Yes | No |
| CopyOnWriteArrayList | Yes | Yes | No | Yes | Yes | Yes | No |
| CopyOnWriteArraySet | Yes | Yes | No | No | Yes | Yes | No |
| HashSet | No | No | No | No | Yes | No | No |
| LinkedHashSet | Yes | No | No | No | Yes | No | No |
| TreeSet | Yes | No | No | No | No | No | No |
| EnumSet | Yes | No | No | No | No | No | No |
| ConcurrentSkipListSet | Yes | No | No | No | No | Yes | No |
| PriorityQueue | Yes | No | No | Yes | No | No | No |
| DelayQueue | Yes | No | No | Yes | No | Yes | No |
| ArrayDeque | Yes | No | No | Yes | No | No | No |
| ArrayBlockingQueue | Yes | No | No | Yes | No | Yes | Yes |
| PriorityBlockingQueue | Yes | No | No | Yes | No | Yes | Yes |
| LinkedBlockingQueue | Yes | No | No | Yes | No | Yes | Yes |
| LinkedBlockingDeque | Yes | No | No | Yes | No | Yes | Yes |
| ConcurrentLinkedQueue | Yes | No | No | Yes | No | Yes | Yes |
| ConcurrentLinkedDeque | Yes | No | No | Yes | No | Yes | Yes |
| LinkedTransferQueue | Yes | No | No | Yes | No | Yes | Yes |
| SynchronousQueue | Yes | No | No | Yes | No | Yes | Yes |
| HashMap | No | No | Yes | No | Yes | No | No |
| LinkedHashMap | Yes | No | Yes | No | Yes | No | No |
| TreeMap | Yes | No | Yes | No | No | No | No |
| WeakHashMap | No | No | Yes | No | Yes | No | No |
| Hashtable | No | No | Yes | No | No | Yes | No |

**Задание 2. Использование Map**

Создать “универсальный” класс, позволяющий получить значение из любого properties-файла. Физическое чтение файла должно происходить только один раз. Результаты чтения храните в коллекции типа Map. Ответьте на вопрос: как ведет себя map-коллекция если в нее добавить элемент с ключом, который уже присутствует?

##### **Задание 3. Ссылки на коллекции**

Определена иерархия классов

**class** MedicalStaff{}

**class** Doctor **extends** MedicalStaff{}

**class** Nurse **extends** MedicalStaff{}

**class** HeadDoctor **extends** Doctor{}

Укажите корректные и некорректные операторы. Дайте ответу пояснение.

|  |  |  |
| --- | --- | --- |
|  | correct | not correct |
| Doctor doctor1 = **new** Doctor(); | + |  |
| Doctor doctor2 = **new** MedicalStaff(); |  | + |
| Doctor doctor3 = **new** HeadDoctor(); | + |  |
| Object object1 = **new** HeadDoctor(); | + |  |
| HeadDoctor doctor5 = **new** Object(); |  | + |
| Doctor doctor6 = **new** Nurse(); |  | + |
| Nurse nurse = **new** Doctor(); |  | + |
| Object object2 = **new** Nurse(); | + |  |
|  |  |  |
| List<Doctor> list1= **new** ArrayList<Doctor>(); | + |  |
| List<MedicalStaff> list2 = **new** ArrayList<Doctor>(); |  | + |
| List<Doctor> list3 = **new** ArrayList<MedicalStaff>(); |  | + |
| List<Object> list4 = **new** ArrayList<Doctor>(); |  | + |
| List<Object> list5 = **new** ArrayList<Object>(); | + |  |

При наследовании используется полиморфизм, в следствии чего можно родительской ссылке присваивать дочерний объект, но нельзя делать в обратную сторону.

Для generic полиморфизм не работает, поэтому можно создавать ссылку и объект только одного типа.

##### **Задание 4. Применение коллекций**

Заполните таблицу.

|  |  |  |
| --- | --- | --- |
|  | Основная функциональность | Примеры типичного использования |
| Set | Проверка наличия элемента во множестве. | Проверка наличия билета с указанным номером. |
| List | Хранение упорядоченного массива объектов с быстрым доступом к любому из объектов. | Хранение списка городов или любых других объектов. |
| Queue | Важность порядка FIFO. | Обработка поступающих запросов на сервер. |
| Map | Попарное хранение объектов, основываясь на паре ключ-значение. | Хранение информации о пользователях, используя в качестве ключа – уникальный никнейм или id пользователя, а в качестве значения – объект User, в котором уже храниться вся информация о самом пользователе. |