Турсунов Баходурхон

**1. Перечислите три основные операции, которые можно выполнить над стеком System.Collections.Generic.Stack<T>, и дайте краткое описание каждой из них.**

Это команды:

Push() – эта операция добавляет элемент в верхушку стека. Элемент, добавленный последним, будет доступен для извлечения первым. Как следствие, стек работает по принципу «последним пришел – первым ушел».

Pop() – эта операция удаляет и возвращает элемент из верхушки стека. Элемент, добавленный последним, будет извлечен первым.

Peek() – эта операция возвращает элемент из верхушки стека без его удаления.

**2. В каких сценариях обычно используется System.Collections.Generic.Stack<T>, в частности для сохранения состояния переменных при вызове процедур?**

Используются для управления вызовами процедур (Call Stack) во время выполнения программы, сохраняя локальные переменные, параметры и адреса возврата для каждой процедуры.

**3. Может ли Stack<T> принимать null в качестве допустимого значения, и допускается ли дублирование элементов?**

Да, Stack<T> в C# может принимать null в качестве допустимого значения.

Что касается дублирования элементов, стандартная реализация Stack<T> в .NET не запрещает дублирования элементов.

**4. Объясните, почему необходимо вызывать метод Sort массива ArrayList перед выполнением операций, требующих его сортировки. Какова альтернатива хранения отсортированной коллекции?**

Вызов метода Sort() для массива ArrayList необходим для упорядочивания его элементов перед выполнением операций, требующих сортировки. Без сортировки операции, такие как поиск, могут быть неэффективными. Альтернативой может быть использование структур данных, таких как SortedSet<T> или SortedDictionary<TKey, TValue>, которые автоматически сортируют элементы при добавлении.

**5. Опишите роль коэффициента загрузки в Hashtable и то, как он влияет на компромисс между временем поиска и потреблением памяти.**

Коэффициент загрузки в Hashtable определяет, насколько таблица заполнена данными относительно ее общей емкости. Он влияет на компромисс между временем поиска и потреблением памяти. Слишком высокий коэффициент загрузки увеличивает вероятность коллизий и замедляет операции, слишком низкий – неэффективно использует память.

**6. Какие требования предъявляются к ключевым объектам в Hashtable в отношении методов Object.GetHashCode и Object.Equals?**

Метод GetHashCode(). Для корректной работы Hashtable объекты-ключи должны правильно реализовывать этот метод, который должен возвращать одинаковое значение для равных объектов. То есть, если два объекта равны согласно методу Equals(), их хэш коды, возвращаемые этим методом, также должны быть равными.

Метод Equals(). Ключевые объекты должны правильно реализовывать этот метод, чтобы определить равенство между ключами. Если два объекта равны согласно этому методу, они должны иметь одинаковый хэш код, возвращаемый методом GetHashCode().

**7. Объясните роль ведер в Hashtable и их вклад в эффективность поиска ключей.**

В Hashtable ведра представляют собой контейнеры, где хранятся элементы по их хэш кодам. Их роль в эффективности поиска заключается в быстром определении местоположения элемента по его ключу. Каждый элемент помещается в соответствующее ведро, что сокращает время доступа к нему, делая операции поиска более эффективными, особенно при больших объемах данных

Practice:

1. Stack:

Напишите программу с использованием Stack<T> для обработки стека операций (например, математических выражений).

Создайте метод для изменения порядка элементов в Stack<T> без использования сторонних коллекций.

2. ArrayList:

Реализуйте программу, использующую ArrayList для хранения данных о студентах (имя, возраст, оценки).

3. Hashtable:

Создайте Hashtable, используя объекты в качестве ключей, и реализуйте методы для добавления, удаления и поиска элементов.