ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР
ПРОГРАММИРОВАНИЯ И ВЫСОКИХ ТЕХНОЛОГИЙ

2020г.

Тип System.Object

- ✓ Основные методы System.Object
- ✓ Сравнение объектов
- ✓ Хэш-код и особенности переопределения методов сравнения
- ✓ Структуры и перечисления
- ✓ Рекурсия
- ✓ Основы отладки кода
- ✓ Упаковка и распаковка

Основные методы System.Object

Все классы в **.NET** являются производными от класса **Object**.

Так же, все значимые типы являются производными от базового класса **ValueType.**

Все типы и классы могут реализовать методы, которые определены в классе **System.Object**.

Метод **Equals(Object)** – определяет, равен ли указанный объект текущему объекту.
Метод **GetHashCode()** – служит в качестве хэш-функции по умолчанию.
Метод **GetType()** – получает тип текущего экземпляра.
Метод **MemberwiseClone()** – создает копию текущего объекта.
Метод **ReferenceEquals(Object, Object)** – определяет, ссылаются ли указанные объекты на один и от же объект в памяти.
Метод **ToString()** – возвращает строковое представление текущего объекта.

Сравнение объектов

ReferenceEquals – сравнивает две ссылки. Если ссылки на объекты идентичны, то возвращает **true**.

В случае передачи этому методу экземпляров значимого типа (даже если передать один и тот же экземпляр) всегда будет возвращать **false**. Так произойдёт потому, что при передаче произойдёт упаковка значимых типов и ссылки на них будут разные.

Здесь также хотелось бы упомянуть о сравнение двух строк этим методом. Он может вернуть **true** и связано это с интернированием строк.

Equals - сначала этот метод проверяет экземпляры на тождество и если объекты не тождественны, то проверяет их на **null** и делегирует ответственность за компарацию переопределяемому экземплярному методу **Equals**. По умолчанию, этот метод ведёт себя точно также как **ReferenceEquals**. Однако для значимых типов он переопределён.

Оператор **==** – для значимых типов всегда следует переопределять, как и виртуальный **Equals()**. Для ссылочных типов лучше не переопределять, ибо, по умолчанию, от **==** на ссылочных типах ожидается поведение как у метода **ReferenceEquals()**.

Хэш-код

Хеширование – преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Результатом хэширования является **хэш-код**.

Хэш-код представляет из себя уникальный идентификатор объекта.

GetHashCode() – в общем и целом, стандартная реализация этого метода ведёт себя как генератор уникального идентификатора. Минус такого подхода состоит в том, что одинаковые семантически объекты, могут возвращать разные hash-значения.

Требования к реализации **GetHashCode():**

- Высокая производительность.
- Хэш-код должен иметь большой разброс в результате.

Структуры и перечисления

Сруктуры:

- Представляют собой ещё один пользовательский тип в **C#.**
- Используется ключевое слово **struct**.
- Являются значимым типом данных.

Как и **class** может использовать конструктор, но это не является обязательным условием. В этом случае необходимо явно инициализировать поля структуры.

```
struct User
    {
        public string name;
        public void DisplayInfo()
        {    }
    }
```

Структуры и перечисления

Перечисления:

- Особый тип данный языка **C#**.
- Представляет набор логически связанных констант.
- Используется ключевое слово **enum**.
- Тип перечисления обязательно представляет целочисленный тип.
- Каждому элементу перечисления присваивается целочисленное значение.

```
enum Sex : byte // имя и тип перечисления
{
    Male,
    Female
}
```

Рекурсия

Рекурсия – это метод, который в ходе своего выполнения вызывает сам себя.

Необходимо предусмотреть условие выхода из рекурсии.

```
static int Factorial(int x)
{
    if (x == 0)
    {
        return 1;
    }
    else
    {
        return x * Factorial(x - 1);
    }
}
```

Основы отладки кода

Если у вас появилась проблема, проанализируйте её и действия которые привели к её появлению:

- Что именно должен был выполнить код?
- Что произошло вместо этого?

Используйте режим пошагового выполнения во время отладки для поиска места возникновения проблемы.

Чтобы перейти в режим отладки в **Visual Studio**, необходимо нажать клавишу **F5** (также вы можете выбрать пункт меню «Отладка» > «Начать» отладку или нажать кнопку «Начать отладку» в панели инструментов «Отладка»).

Если возникает исключение, помощник по исправлению ошибок **Visual Studio** направит вас к точке его появления и предоставит другую необходимую информацию.

Стратегия поиска ошибок

- Ручная отладка кода при помощи точек останова.
- Использование конструкции **try...catch...finally.**

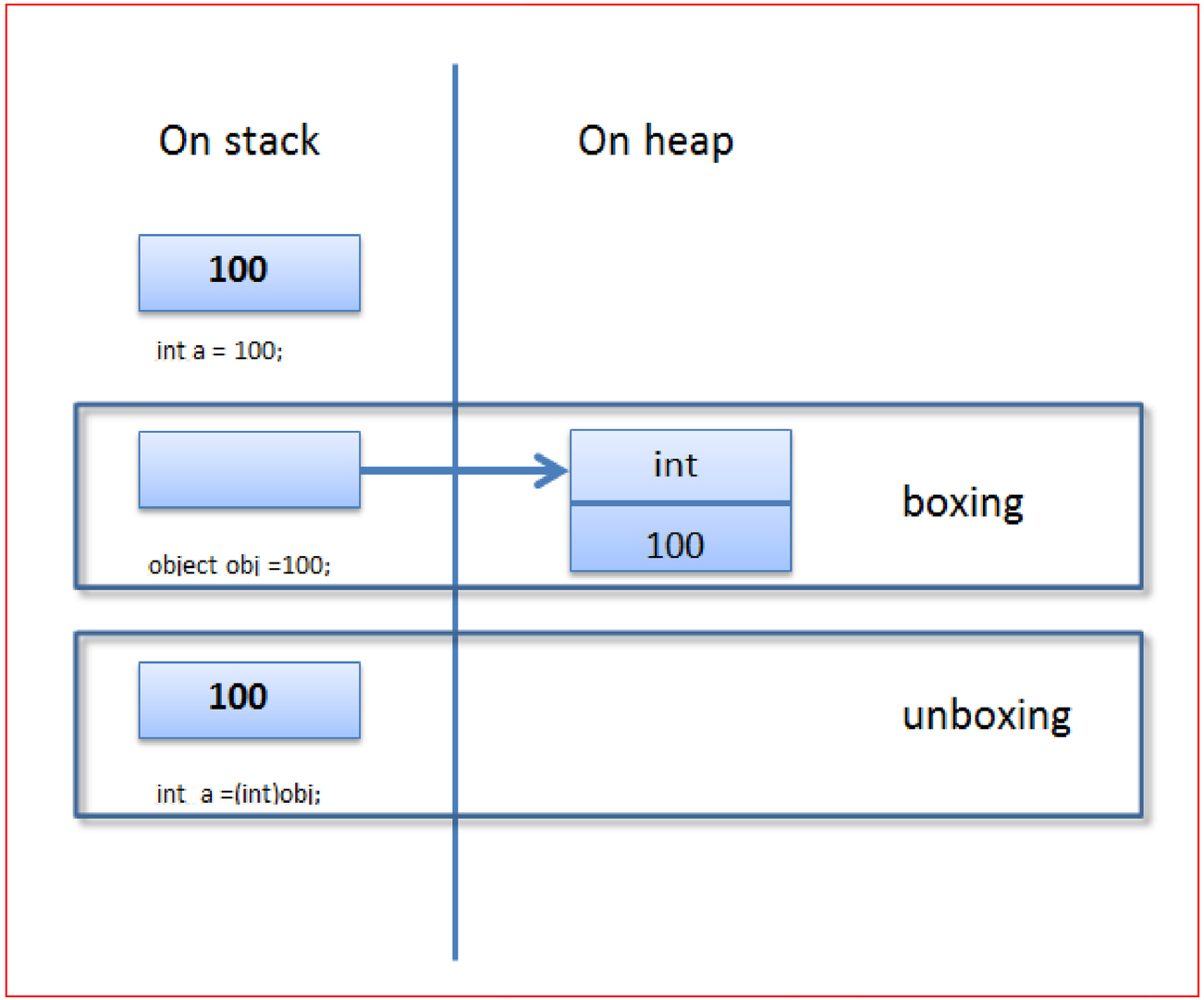
```
try
{
    // блок кода в котором ожидантся исключение
}
catch
{
    // блок кода для обработки исключения
}
finally
{
    // блок кода который выполнится в любом случае
}
```

ФОРМЫ ЗАПИСИ АЛГОРИТМОВ

Так как **System.Object** является предком любого типа, переменной типа **object** можно присвоить любую переменную.

Для типов значений выполняется специальная операция, называемая операцией упаковки (**boxing**).

Обратное преобразование – операция распаковки (**unboxing**).



Задание

1. Создать тип Car с двумя полями: int Age и Color Color. Color – это enum. Сделать так, чтобы при равных Age и Color, операция == и Equals() возвращалось True.

2. Добавить типу Car возможность сортироваться, если он содержится в коллекции.

3. Сделать так, чтобы условный оператор If(Car car) возвращал true, если Age > 4, иначе false.