Языки C# и Java появились в разное время. Язык Java был создан задолго до появления C#. Под названием OakJava был разработан компанией Sun в 1990 г., а в 1995 была выпущена первая бета-версия Java. Создание C# было анонсировано в 2000 году, а в 2002 году вышла первая версия платформы .NET, поддерживающей C#. Таким образом, если Java создавался опираясь в большей степени на опыт языков Objective C и C, то для C# такой опорой являлись C++ и сам Java[1]. И, несмотря на своё название, C# оказался ближе к Java, чем к C++[2][3].

С точки зрения разработчика языки Java и C# очень похожи. Оба языка являются строго типизированными, объектно-ориентированными. Оба вобрали в себя многое из синтаксиса C++, но в отличие от C++, проще в освоении для начинающих. Оба языка используют сборку мусора. Оба используют фигурные скобки для выделения блоков. Оба языка сопровождаются богатыми коллекциями библиотек. Но есть в языках также свои особенности и различия, сильные и слабые стороны. C# учёл многие недостатки Java, и исправил их в своей реализации[4]. Но и Java не стоит на месте, развиваясь параллельно с C#.

Кик Рэдек из Microsoft считает C# более сложным языком, чем Java[1]. По его мнению, «язык Java был построен таким образом, чтобы не дать возможности разработчику прострелить себе ногу» (англ. *«Javawasbuilttokeepadeveloperfromshootinghimselfinthefoot»*), а «C# был построен так, чтобы дать разработчику ружьё, но оставить его на предохранителе» (англ. *«C# was built to give the developer a gun but leave the safety turned on»*).

Оба языка используют в качестве синтаксической основы язык программирования C. В частности, от него унаследованы без изменений:

- обозначения начала/конца блока кода фигурными скобками;
- обозначения, ассоциативность и приоритет большинства встроенных операций (присвоение, арифметические, логические, побитовые операции, операции инкремента/декремента, тернарная условная операция«**?:**»);
- синтаксис описания и использования переменных и функций (порядок «тип имя», использование модификаторов, обязательность скобок для функций, описание формальных параметров);
- синтаксис всех основных конструкций: условного оператора, циклов, оператора множественного выбора;
- Оба языка реализуют одну модель работы с динамическими данными: объекты создаются динамически с помощью конструкции `new`, среда исполнения отслеживает наличие ссылок на них, а сборщик мусора периодически очищает память от объектов, ссылок на которые нет. Для оптимизации сборки мусора спецификации языков и сред исполнения не содержат ограничений на время жизни объекта после удаления последней ссылки на него — сборщик работает независимо от исполнения программы, поэтому реальное уничтожение объекта может произойти в любой момент после удаления последней ссылки до завершения работы программы. В реальности сборщики мусора оптимизируют исполнение так, чтобы обеспечить приемлемый расход памяти при минимальном замедлении работы программ.
- И в Java, и в C# есть сильные и слабые ссылки на объекты. Оба языка поддерживают методы-финализаторы. Из-за неопределённости момента удаления объекта финализаторы не могут использоваться для освобождения системных ресурсов, занятых объектом, что вынуждает создавать дополнительные методы для «очистки» объекта и вызывать их явно.

- **Объектные средства**
- Оба языка — объектно-ориентированные, с синтаксисом, унаследованным от C++, но значительно переработанным. Код и данные могут описываться только внутри классов.

Инкапсуляция

В Java модификатор protected в описании, помимо доступа из классов-потомков, разрешает доступ из всех классов, входящих в тот же пакет, что и класс-владелец.

В C# для объектов, которые должны быть видны в пределах сборки (примерный аналог пакета Java) введён отдельный модификатор internal, а protected сохраняет свой изначальный смысл, взятый из C++ — доступ только из классов-потомков. Допускается комбинировать internal и protected — тогда получится область доступа, соответствующая protected в Java.