

# РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Перенос кода

# МЕТОДЫ ПЕРЕНОСА КОДА

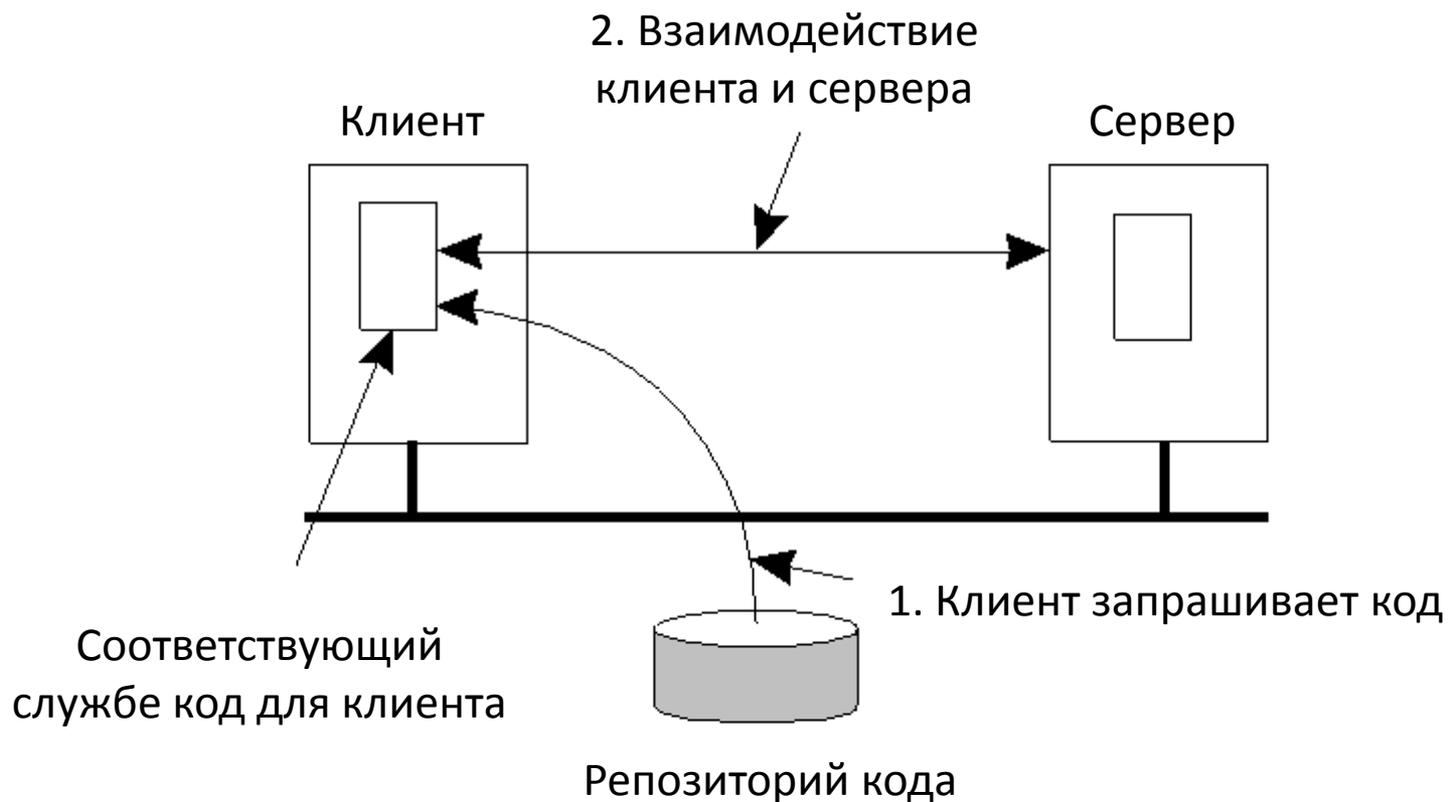
# ПРИНЦИПЫ ДЕЦЕНТРАЛИЗАЦИИ

- ◎ Как известно, один из принципов компьютерной техники – единство представления данных и кода.
- ◎ В связи с этим возникает возможность переноса не только данных приложения, но и исполняемого кода на удаленную машину.

# ОСНОВНЫЕ ПРЕИМУЩЕСТВА ПЕРЕНОСА КОДА

- ◎ Балансировка нагрузки вычислительных модулей распределенной ВС и повышение производительности.
- ◎ Уменьшение трафика в коммуникационной подсистеме распределенной ВС.
- ◎ Обеспечение гибкости распределенных ВС и возможности динамического конфигурирования.
- ◎ Отсутствие необходимости иметь заранее сконфигурированное прикладное ПО в узлах системы.

# ГИБКОСТЬ



Перенос кода в PWS происходит в форме миграции процессов.

В процессе можно выделить следующие сегменты:

- 1. Сегмент кода:** исполняемый код программы
- 2. Сегмент ресурсов:** содержит ссылки на внешние ресурсы, используемые процессом
- 3. Сегмент исполнения:** хранение состояния процесса (стек, счетчик программы и т.п.)

# СЛАБАЯ МОБИЛЬНОСТЬ

- ◎ **Слабая мобильность** (weak mobility) – перенос только *сегмента кода* (возможно с некоторыми данными для инициализации).
- ◎ Перенесенная программа всегда запускается с исходного состояния.
- ◎ Пример – Java-апплеты.
- ◎ Достоинства: простота реализации

# СИЛЬНАЯ МОБИЛЬНОСТЬ

- ◎ **Сильная мобильность (strong mobility)** – перенос сегмента кода и сегмента исполнения.
- ◎ Исполнение процесса можно начать с того места, на котором он был остановлен на предыдущей машине.
- ◎ Возможен как перенос процесса (с уничтожением на источнике), так и его клонирование

# МЕТОДЫ ИНИЦИАЦИИ ПЕРЕНОСА КОДА

9

Перенос кода может быть инициирован:

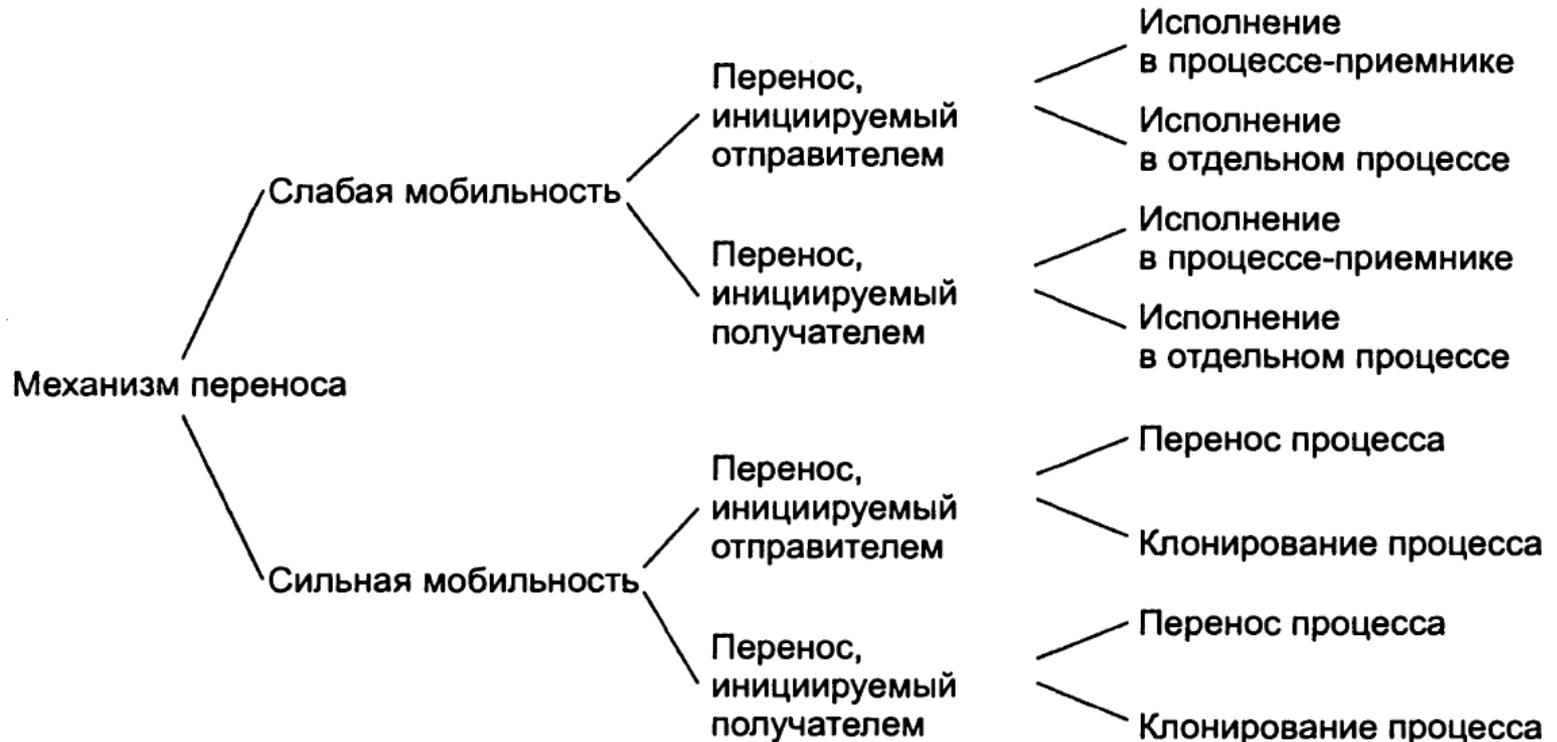
- ⊙ Отправителем: передача поисковых программ на сервер баз данных, перенос и запуск троянских программ (серьезные проблемы в безопасности);
- ⊙ Получателем: Java-апплеты (проблемы в безопасности тоже весомые, но не так очевидны как в первом случае).

# МЕТОДЫ ЗАПУСКА ПЕРЕНЕСЕННОГО КОДА

Переносимый код может выполняться в процессе-приемнике или же в новом, отдельном процессе:

- ⊙ Исполнение в процессе-приемнике (Java-апплет; требует отдельной защиты процесса-приемника от возможного вредоносного кода)
- ⊙ Исполнение в отдельном процессе

# МОДЕЛИ ПЕРЕНОСА КОДА



# ПЕРЕНОС КОДА И ЛОКАЛЬНЫЕ РЕСУРСЫ

# СВЯЗЬ ПРОЦЕССА С ЛОКАЛЬНЫМИ РЕСУРСАМИ

В отличие от сегмента кода и сегмента исполнения **сегмент ресурсов** невозможно также легко перенести на удаленную машину.

Выделяют 3 типа привязки ресурса к локальному процессу:

- ◎ **По идентификатору** – процесс требует в точности тот ресурс, на который ссылается (URL-адрес для ссылки на конкретный документ);
- ◎ **По значению** – процессу необходимо только значение ресурса (стандартные библиотеки Java)
- ◎ **По типу** – при обращении процесса к локальным устройствам (экран, принтер и т.п.)

# ПРИВЯЗКА РЕСУРСА К КОНКРЕТНОМУ УЗЛУ

По способу привязки к конкретному узлу ВС ресурсы делятся на:

- ◎ **неприсоединенные**, т.е. легко переносимые с одного узла ВС на другой (например, файлы данных);
- ◎ **связанные**, перенос которых сопряжен с большими трудностями (например, базы данных или веб-сайты целиком);
- ◎ **фиксированные**, которые изначально привязаны к конкретной машине и не могут быть перенесены на другую (например, локальные устройства).

# ВОЗМОЖНЫЕ ВАРИАНТЫ ПЕРЕНОСА РЕСУРСОВ

Выделяют 4 варианта решения задачи переноса локальных ресурсов:

1. **GR** (global reference )- организация ссылки;
2. **MV** (move) - перенос ресурса;
3. **CP** (copy) - копирование значения ресурса;
4. **RB** (rebind) - выполнение новой привязки к локальному ресурсу.

# РЕШЕНИЕ ЗАДАЧИ ПЕРЕНОСА РЕСУРСОВ

16

	Неприсоединенный	Связанный	Фиксированный
По идентиф.	MV (или GR)	GR (или MV)	GR
По значению	CP ( или MV, GR)	GR (или CP)	GR
По типу	RB (или GR, CP)	RB (или GR, CP)	RB (или GR)

- ◎ **По идентификатору:** лучше всего перенести его вместе с кодом. Однако, если он используется совместно с другими процессами – то лучше организовать глобальную ссылку.

# РЕШЕНИЕ ЗАДАЧИ ПЕРЕНОСА РЕСУРСОВ

	Неприсоединенный	Связанный	Фиксированный
По идентиф.	MV (или GR)	GR (или MV)	GR
По значению	CP ( или MV, GR)	GR (или CP)	GR
По типу	RB (или GR, CP)	RB (или GR, CP)	RB (или GR)

- ◎ **По значению:** обычно это библиотеки времени использования. Они допускают копирование. Если же копирование недопустимо (большой объем), возможно организовать глобальную ссылку.

# РЕШЕНИЕ ЗАДАЧИ ПЕРЕНОСА РЕСУРСОВ

	Неприсоединенный	Связанный	Фиксированный
По идентиф.	MV (или GR)	GR (или MV)	GR
По значению	CP ( или MV, GR)	GR (или CP)	GR
По типу	RB (или GR, CP)	RB (или GR, CP)	RB (или GR)

- ◎ **По типу:** независимо от способа привязки, лучший способ – это привязка процесса к аналогичным ресурсам на локальной машине. Если это невозможно – то желательно постараться сделать глобальную ссылку на них.

# ПЕРЕНОС КОДА В ГЕТЕРОГЕННЫХ СРЕДАХ

# ПРОБЛЕМА ПЕРЕНОСА КОДА В ГЕТЕРОГЕННЫХ СРЕДАХ

- ◎ В гомогенных системах перенос кода очевиден
- ◎ В гетерогенных можно решить проблему переноса кода (слабая мобильность) создав по одному варианту исполняемого кода на каждый тип ВС.
- ◎ Но как предугадать появление новой платформы?
- ◎ Как организовать сильную мобильность?

# КОНЦЕПЦИЯ ВИРТУАЛЬНЫХ МАШИН

21

Виртуальная машина –

- ⊙ это программная или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы и исполняющая программы для этой платформы (целевая платформа) на другой платформе (хост-платформа)
- ⊙ или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы (песочница);

# ИСПОЛНЯЕМЫЙ КОД ВМ

В зависимости от типа, ВМ может исполнять:

- ◎ некоторый машинно-независимый код:
  - ◎ *байт-код Java* выполняется *виртуальной машиной Java*;
  - ◎ *ActionScript* выполняется *виртуальной машиной FlashPlayer*;
  - ◎ *IntermediateLanguage* выполняется *.NET Common Language Runtime*;
- ◎ или машинный код реального процессора.

# СРЕДА JAVA (JVM)

Идея Java зародилась в 1980 (язык Oak («дуб») для программирования бытовых электронных устройств).

Назван в честь марки кофе Java, любимого некоторыми программистами.

Основные компоненты среды Java:

- ☉ язык программирования Java;
- ☉ платформа Java.

# ТИПЫ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ JAVA

- ◎ **приложения** – программы в обычном смысле, выполняемые, однако, в среде платформы Java;
- ◎ **апплеты** – программы, выполняемые в среде Web-браузера, поддерживающего платформу Java;
- ◎ **сервлеты** – Java-программы, серверные компоненты распределенных приложений;
- ◎ **программы, выполняющиеся в средах продуктов** промежуточного программного обеспечения (Oracle и т.п.)

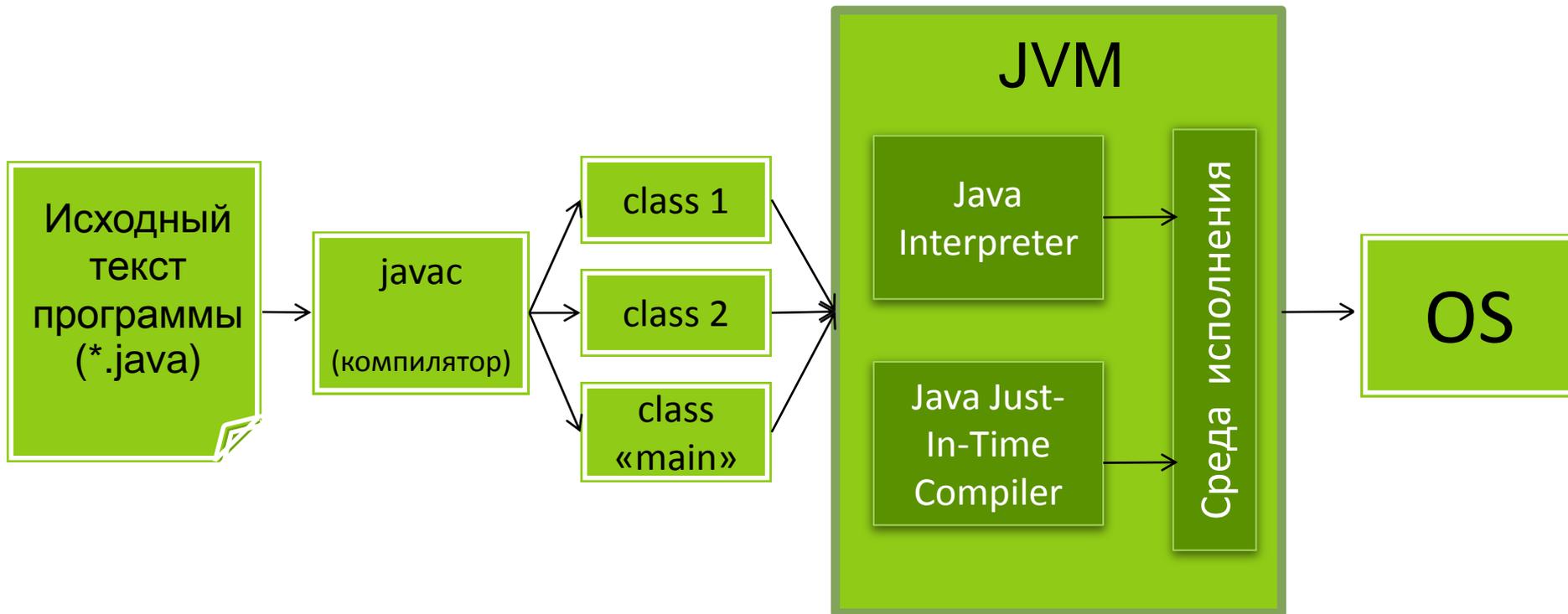
# ВИРТУАЛЬНАЯ МАШИНА

## JAVA

25

- ◎ **Переносимость в Java** достигается за счет того, что Java-программа компилируется в так называемый **байт-код**.
- ◎ **Байт-код** – Java - команды некоторой абстрактной машины, называемой **виртуальной машиной Java (JVM)**
- ◎ Байт-код не зависит от базовой операционной системы и оборудования ВС.

# ИСПОЛНЕНИЕ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ JAVA

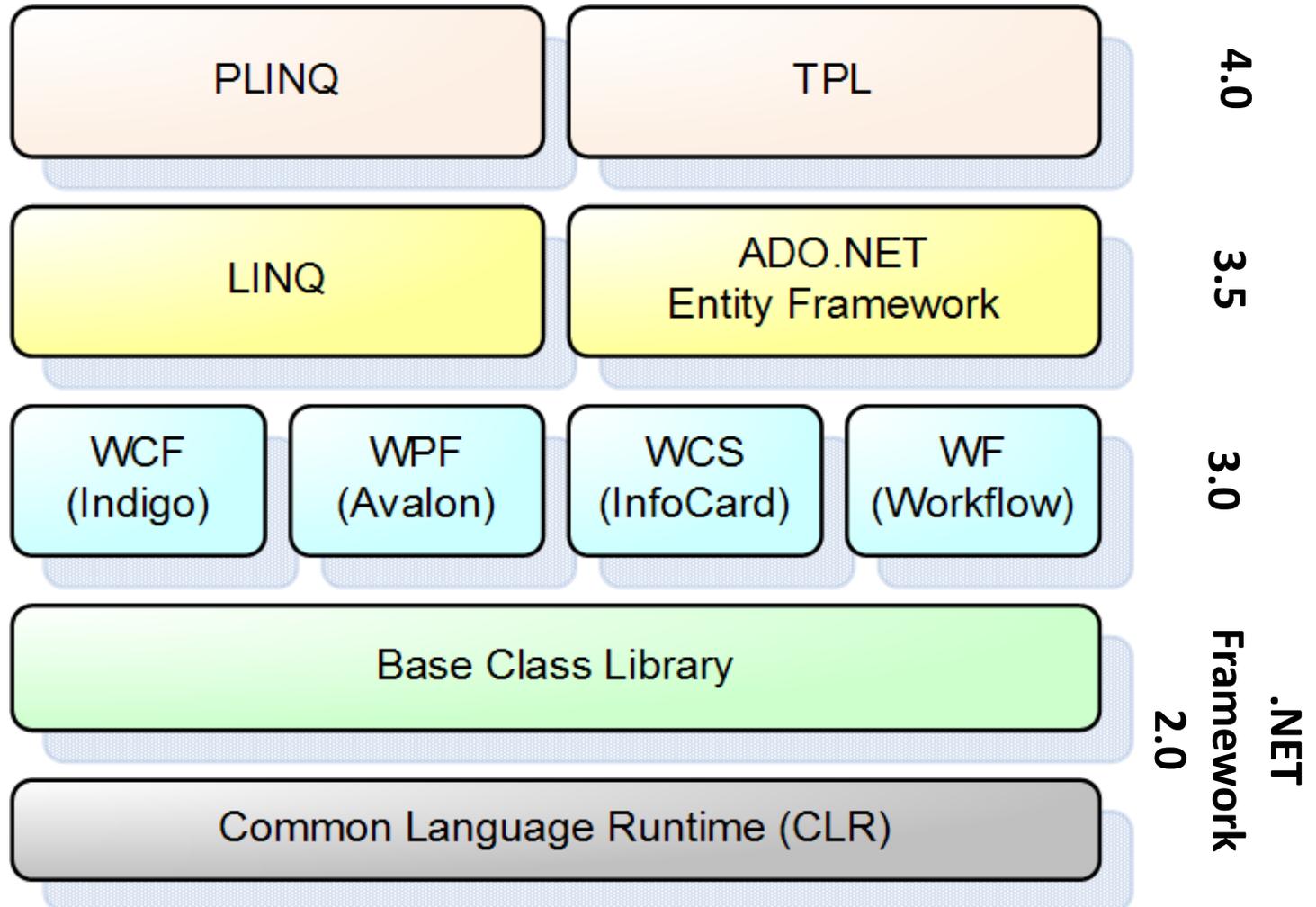


# ИНФРАСТРУКТУРА .NET FRAMEWORK

- ◎ .NET Framework – это программная инфраструктура от компании Microsoft, обеспечивающая разработку как настольных, так и web-приложений.
- ◎ Основная идея .NET – совместимость программных систем, написанных на разных языках (на основе трансляции всех программ в байт-код **CIL - Common Intermediate Language**).

# СТЕК ТЕХНОЛОГИЙ .NET FRAMEWORK

28



# КРОСПЛАТФОРМЕННОСТЬ .NET FRAMEWORK

29

- ◎ .NET изначально не поддерживал кроссплатформенность.
- ◎ Mono — проект по созданию полноценного воплощения системы .NET на базе свободного программного обеспечения.
- ◎ Объем реализации:
  - ◎ компилятор C#;
  - ◎ среда исполнения .NET
  - ◎ отладчик;
  - ◎ блок библиотек (вплоть до .NET 3.5 кроме WPF и WF)