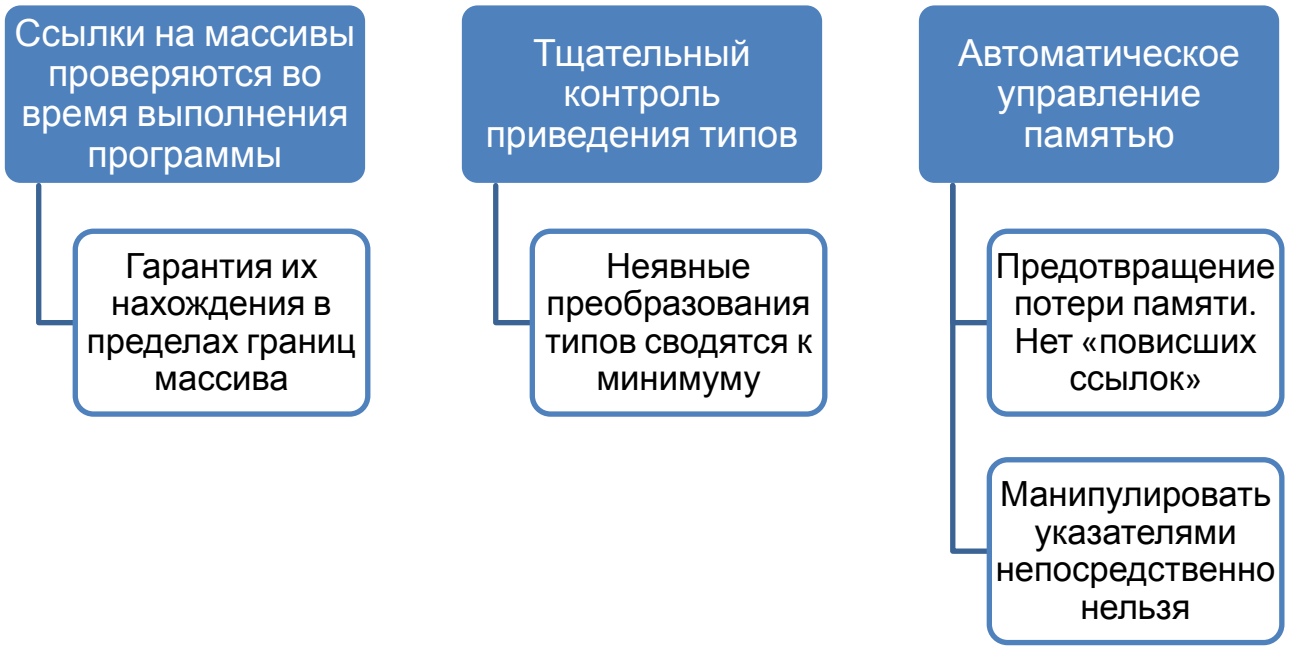
Лекция 3. Модели безопасности в программировании

- Модель безопасности данных в Java
- «Песочница» Java
- Организация защиты на платформе Java

Модель безопасности данных в Java

Модель безопасности языка Java состоит из трех уровней, причем каждый из них зависит от нижестоящих уровней. Первый уровень защиты от недостоверных программ в приложении языка Java составляет часть основной конструкции этого языка, поскольку Java является безопасным языком. Безопасность обеспечивается несколькими путями:



Модель безопасности данных в Java

Второй уровень безопасности языка Java подразумевает тщательную проверку правильности файлов классов языка Java при их загрузке в виртуальную машину (включая байтовые коды виртуальной машины, которые представляют собой скомпилированные варианты методов). Эта проверка гарантирует, что испорченный файл класса не сможет обусловить появление ошибки в самом интерпретаторе языка Java, и вместе с тем защищает от нарушения основной системы безопасности языка.

Модель безопасности данных в Java

Третьим и последним уровнем модели безопасности в языке Java является реализация библиотеки класса языка Java. Классы в библиотеке языка Java предоставляют приложениям лишь собственные средства доступа к уязвимым системным ресурсам, например к файлам и сетевым соединениям. Эти классы написаны таким образом, чтобы всегда выполнять проверку безопасности до предоставления доступа.

Модель безопасности данных в Java

1. Часть основной конструкции языка
2. Проверка правильности файлов классов языка Java при их загрузке в виртуальную машину
3. Приложениям предоставляются только собственные средства доступа к уязвимым системным ресурсам.

Иными словами, надежность языка, контроль при получении и загрузке программ, контроль при выполнении программ.

«Песочница» Java.

Java представляет собой очень мощный язык разработки. Ненадежным апплетам не должно быть позволено получить доступ ко всей этой мощи. Java песочница ограничивает апплеты от выполнения многих мероприятий.

Безопасности Java основана на трех столбах обороны: Верификатор Байт кода, загрузчик класса, и менеджер по безопасности (Class Loader, Byte Code Verifier and Security Manager).

«Песочница» Java. Class Loader

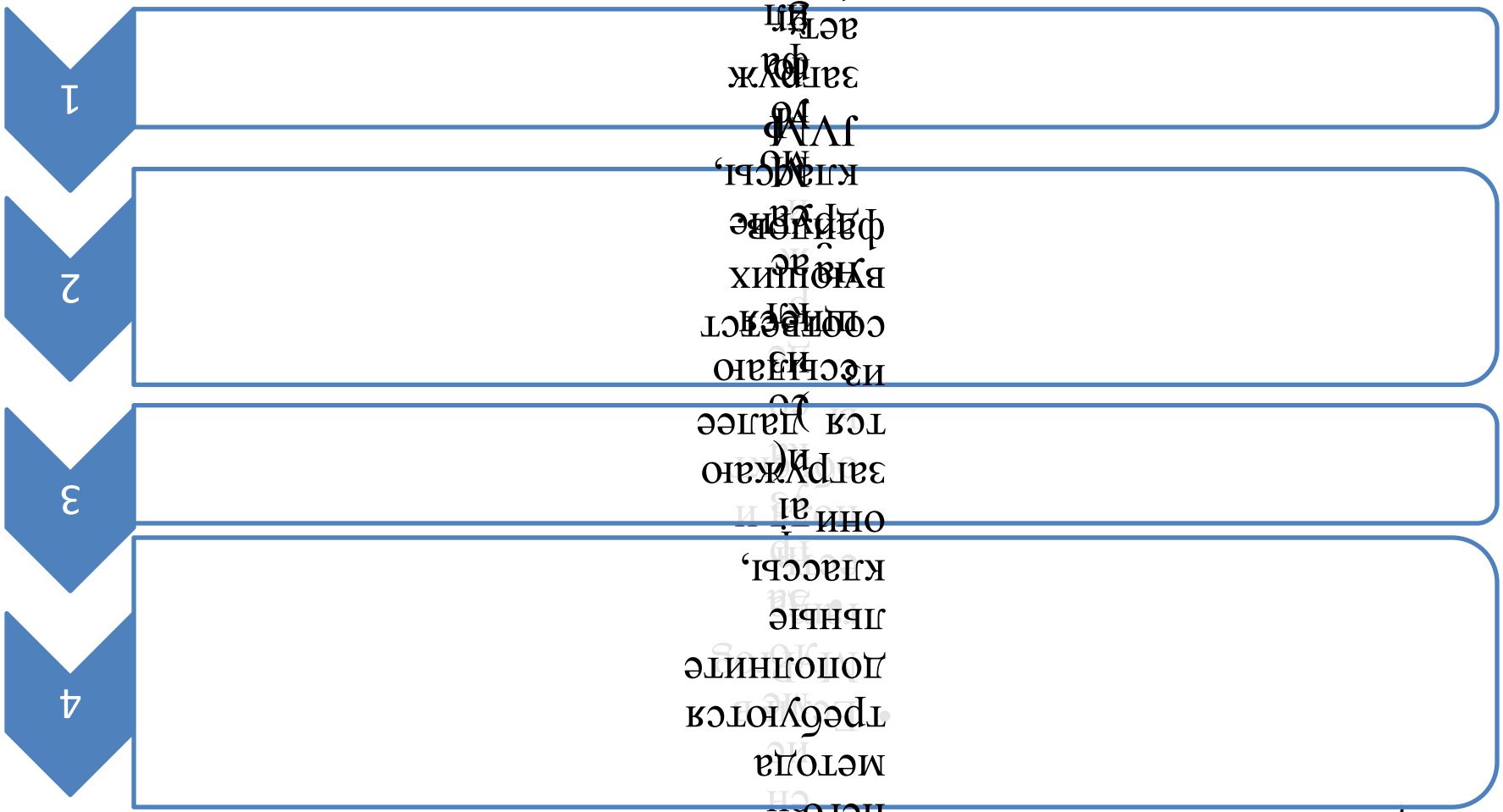
Компилятор Java преобразует исходные операторы языка в понятный для виртуальной машины Java байт-код, который сохраняется в файле класса с расширением .class.

В каждом файле класса содержится код определения и реализации только для одного класса или интерфейса. Далее все эти файлы классов интерпретируются программой, которая способна преобразовывать инструкции виртуальной машины в машинные команды целевой платформы.

Следует иметь ввиду, что интерпретатор виртуальной машины загружает только те файлы классов, которые потребуются для выполнения программы в данный момент.

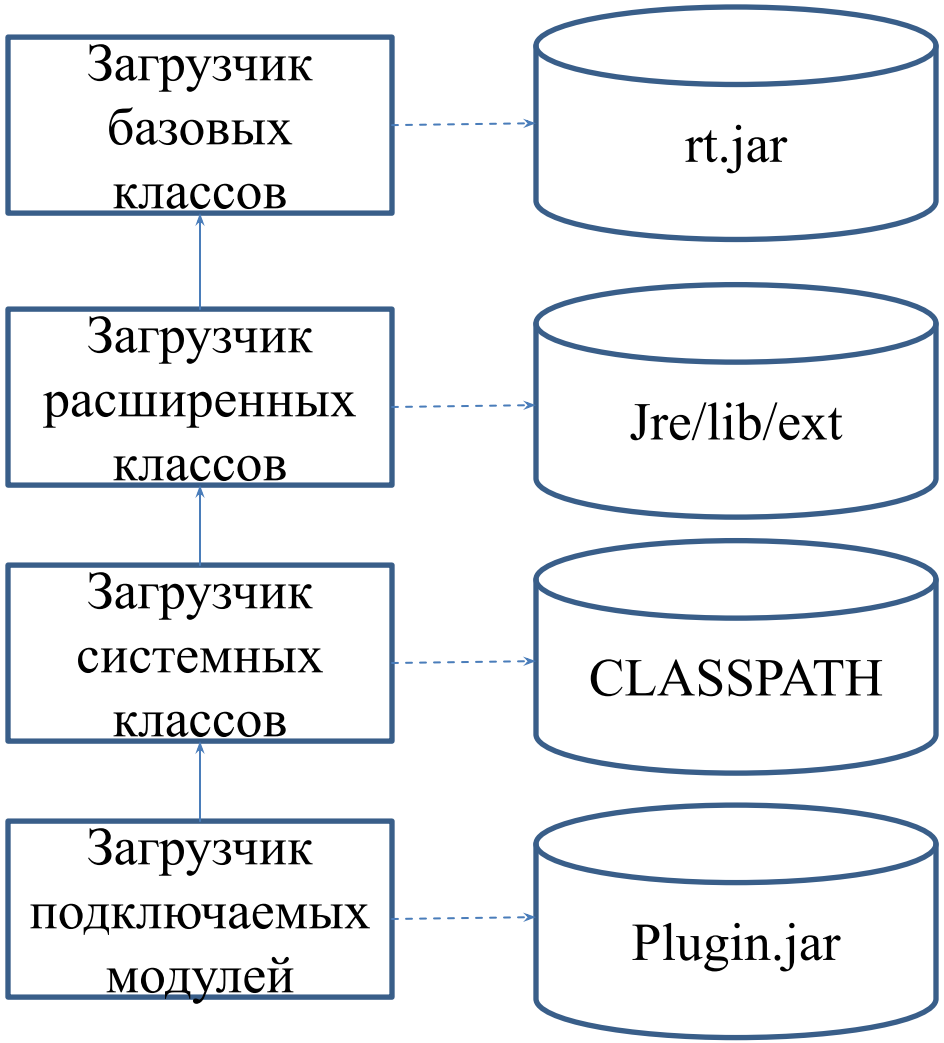
«Песочница» Java. Class Loader

Действия JVM при запуске программы, выполнение которой начинается с файла MyProgram.class



«Песочница» Java. Class Loader

Иерархия загрузчиков классов.



Java

«Песочница» Java. Byte Code Verifier

Верификация байт-кода. Когда загрузчик классов представляет виртуальной машине байт-код класса, этот код сначала обследуется верификатором. Верификатор проверяет все классы за исключением системных и определяет те команды, которые могут нанести ущерб виртуальной машине.

Виды проверок, выполняемых верификатором:

- Инициализация переменных перед использованием
- Согласование типов ссылок при вызове метода
- Соблюдение правил доступа к закрытым данным и методам
- Доступ к локальным переменным в стеке во время выполнения
- Отсутствие переполнения стека

При невыполнении какой-нибудь из этих проверок класс считается поврежденным и не загружается.

«Песочница» Java. Byte Code Verifier

Невыполнение какой-либо из этих проверок:

- Подделка указателей (например, получение указателя как результат выполнения арифметической операции);
- Вызов методов объектов с недопустимым набором параметров;
- Недопустимое преобразование типов;
- Вызов операции Java-машины с недопустимым набором параметров;
- Некорректная операция с регистрами Java-машины (например, запись регистра с неопределенным содержимым);

Также влечет отказ в загрузке класса, а сам класс считается поврежденным.

«Песочница» Java. Byte Code Verifier

Java байт-код "проверяется", прежде чем он сможет заработать. Эта схема проверки предназначена для того, чтобы байт-код, который может был, а может и не был создан с помощью компилятора Java, играл по правилам. В конце концов, байт-код возможно был создан во "враждебном" компиляторе, чей байт-код, предназначенный для сбоя виртуальной машины Java.

Верификатор проверяет байт-код на различных уровнях. Простейший тест **гарантирует, что формат фрагмента байт-кода правильный**. Более глубоком уровне, встроенные доказательства теорем применяются **к каждому фрагменту**. Теорема помогает убедиться, что байт-код не подделывает указатели, не нарушает ограничения доступа, или доступ к объектам, используя информацию неправильного типа.

Процесс проверки, совместно с функциями безопасности, встроенные в язык через компилятор, помогает **установить базовый набор гарантий безопасности.**

«Песочница» Java. Security Manager

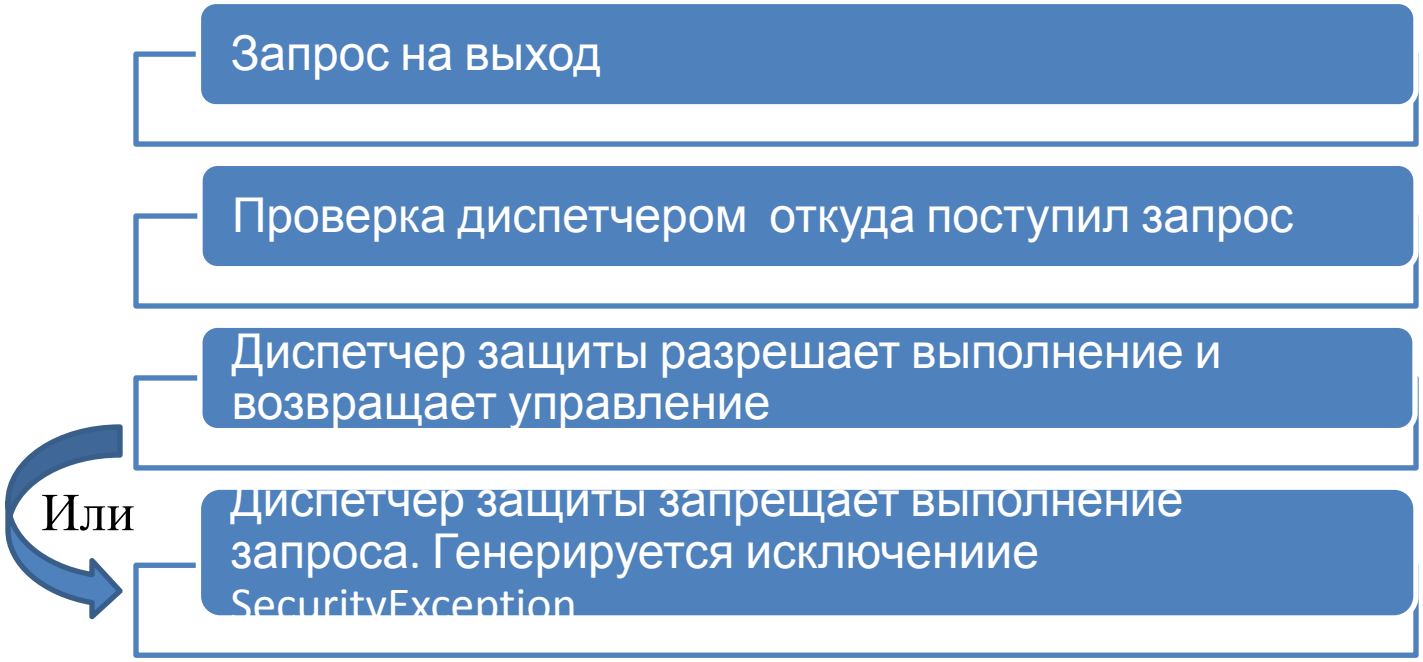
После загрузки класса в виртуальную машину и проверки верификатором в действие вступает еще один механизм обеспечения безопасности в Java: диспетчер защиты. В качестве диспетчера защиты служит класс, определяющий, разрешено ли коду выполнять ту или иную операцию. Операции, попадающие под контроль диспетчера защиты:

- Создание нового загрузчика классов
- Выход из виртуальной машины
- Получение доступа к члену другого класс с помощью рефлексии
- Получение доступа к файлу
- Установление соединения через сокет
- Запуск задания на печать
- Получение доступа к системному буферу обмена
- Получение доступа к очереди событий в AWT
- Обращение к окну верхнего уровня

«Песочница» Java. Security Manager

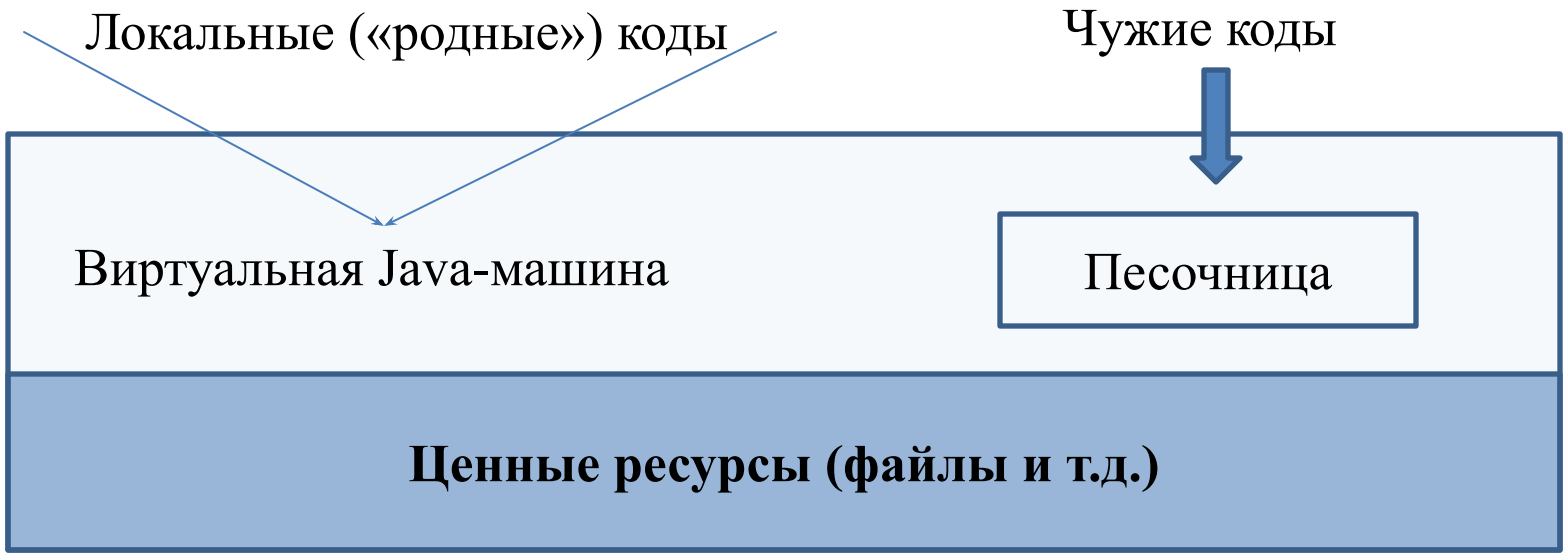
Программа просмотра аплетов принудительно применяет правила защиты, которые в значительной степени ограничивают количество разрешенных операций.

Например, аплетам не разрешается завершать работу виртуальной машины. При попытке аплета вызвать метод exit() выдается исключение системы безопасности.



Организация защиты на платформе Java

Основная цель мер безопасности в **Java** — обеспечить защиту **Java**-окружения от вредоносных программ. Для достижения этой цели в JDK 1.0 была предложена концепция "песочницы" (sandbox) — замкнутой среды, в которой выполняются потенциально ненадежные программы. Таковыми считались аплеты, поступившие по сети. Весь "родной" код (то есть программы, располагающиеся на локальном компьютере) считался абсолютно надежным и ему было доступно все, что доступно виртуальной **Java**-машине.



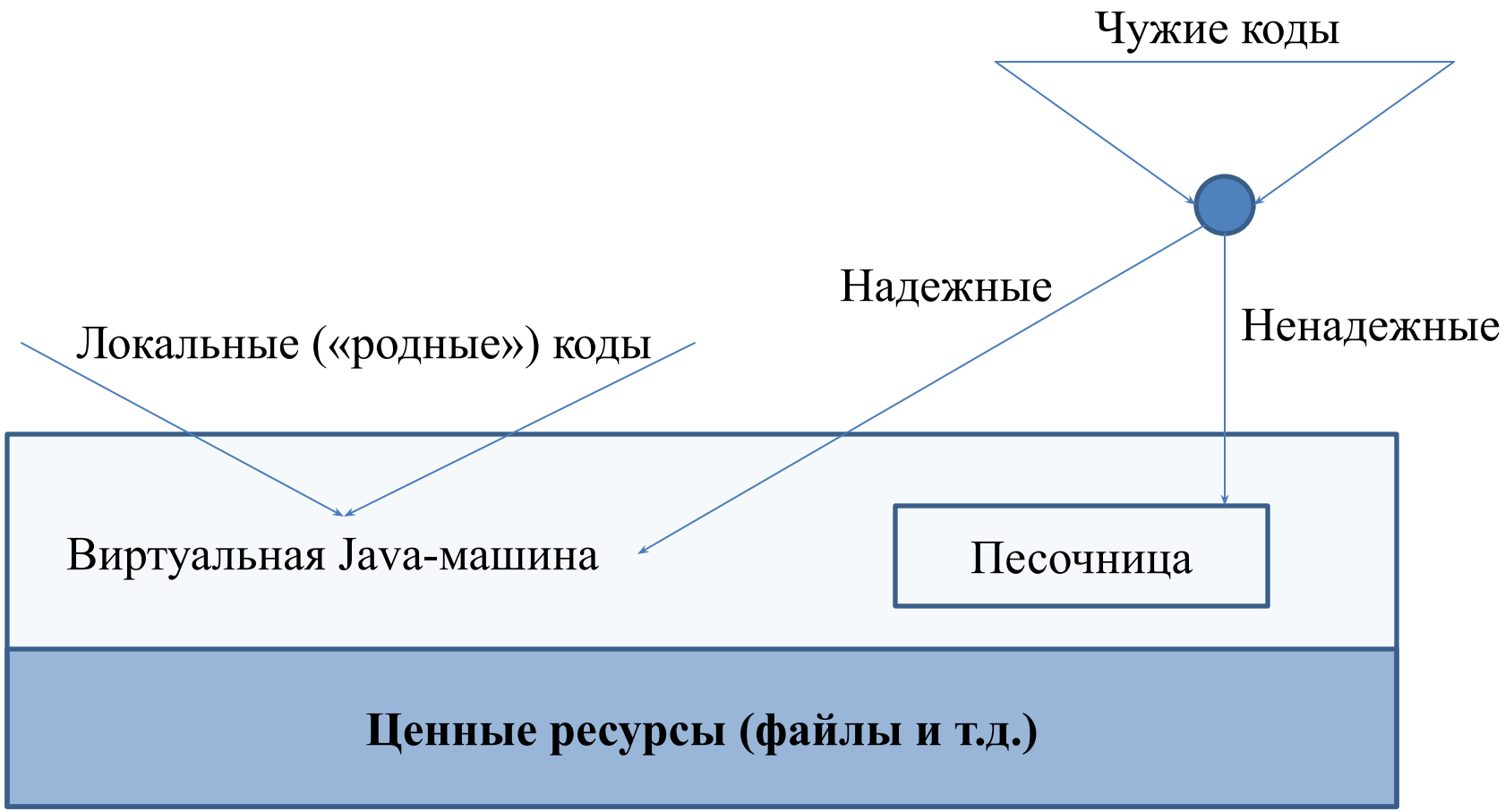
Организация защиты на платформе Java

В число ограничений, налагаемых "песочницей", входит **запрет на доступ к локальной файловой системе**, на сетевое взаимодействие со всеми хостами, кроме источника аплета (хост, с которого аплет был получен) и т.д. При таких ограничениях безопасность в общем и целом обеспечивается, но возможности для работы у аплетов почти не остается.

Чтобы как-то справиться с этой проблемой, в **JDK 1.1** ввели **понятие электронной подписи**, которую ставит распространитель аплета. **Java**-машина в соответствии со своей политикой безопасности делит распространителей и, соответственно, их аплеты на две категории — **надежные и ненадежные** (неподписанный аплет, естественно, считается ненадежным).

Надежные аплеты были приравнены в правах к "родному" коду, в результате чего **модель безопасности эволюционировала** к виду, приведенному на следующем слайде.

Организация защиты на платформе Java



Организация защиты на платформе Java

В JDK 1.2 по существу произошло обобщение «песочницы». Оформились три основных понятия:

- Источник программы
- Право и множество прав
- Политика безопасности

Источник программы определяется парой (универсальный локатор ресурсов — URL, распространители программы — те, кто подписал ее). URL может указывать на файл в локальной файловой системе или же на ресурс удаленной системы.

Организация защиты на платформе Java

Право — это абстрактное понятие, за которым, как обычно в **Java**, стоят классы и объекты. В большинстве случаев право определяется двумя цепочками символов — именем ресурса и действием.

Например, в качестве ресурса может выступать файл, а в качестве действия — чтение. Важнейшем методом "правовых" объектов является implies(). Он проверяет, что одно право (запрашиваемое) следует из другого (имеющегося).

Политика безопасности задает соответствие между источником и правами поступивших из него программ. Вообще говоря, в JDK 1.2 "родные" программы не имеют каких-либо привилегий в плане безопасности и политика по отношению к ним может быть любой.

Организация защиты на платформе Java

По сути мы имеем традиционный для современных операционных систем и систем управления базами данных механизм прав доступа со следующими особенностями:

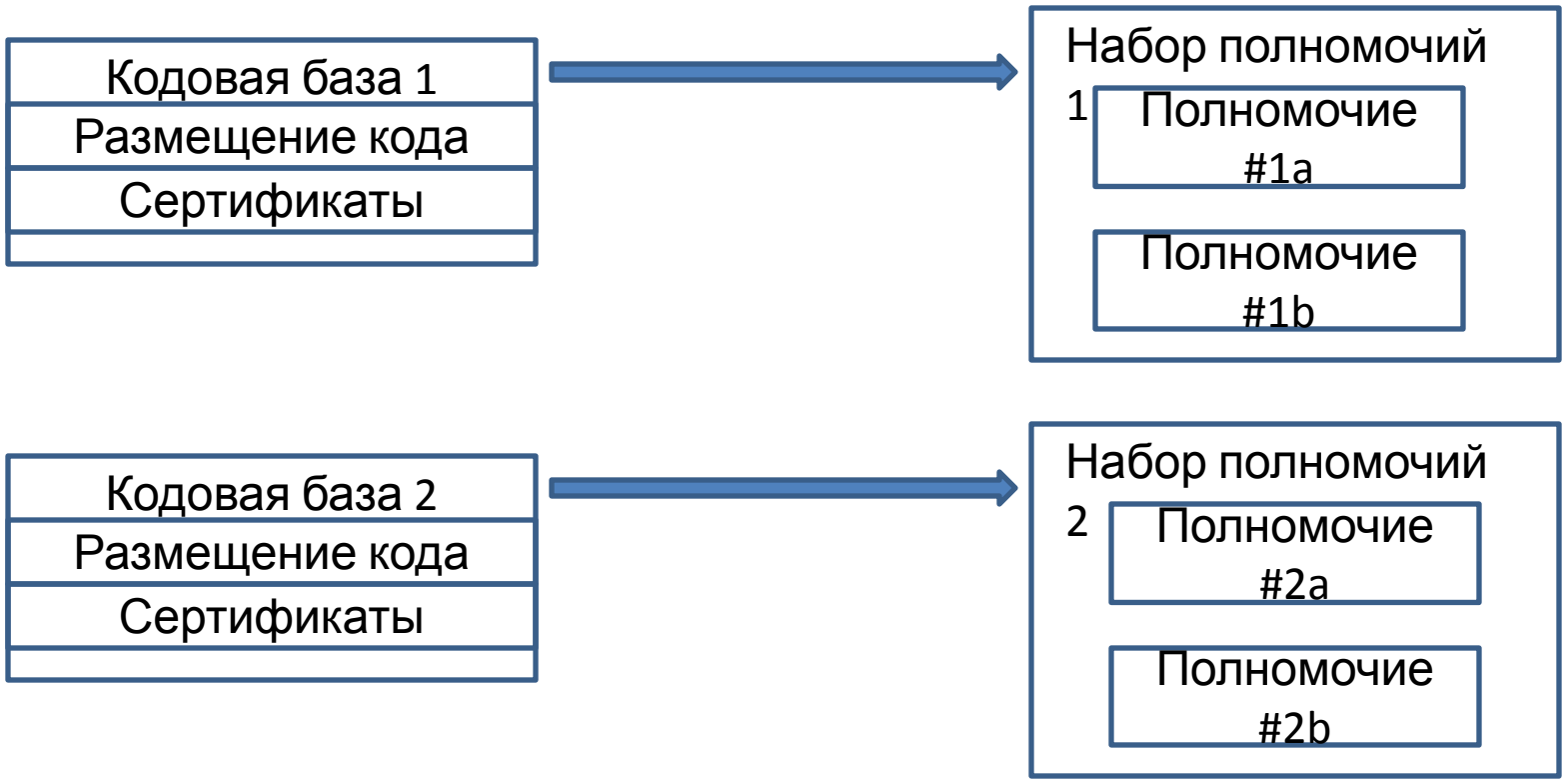
- Субъектом доступа является не пользователь, а источник программы. Впрочем, формально можно считать, что во время исполнения программы ее источник становится пользователем

- Нет понятия владельца ресурса, который (владелец) мог бы менять права; последние задаются исключительно политикой безопасности. Впрочем, формально можно считать, что владельцем всего является тот, кто формирует политику

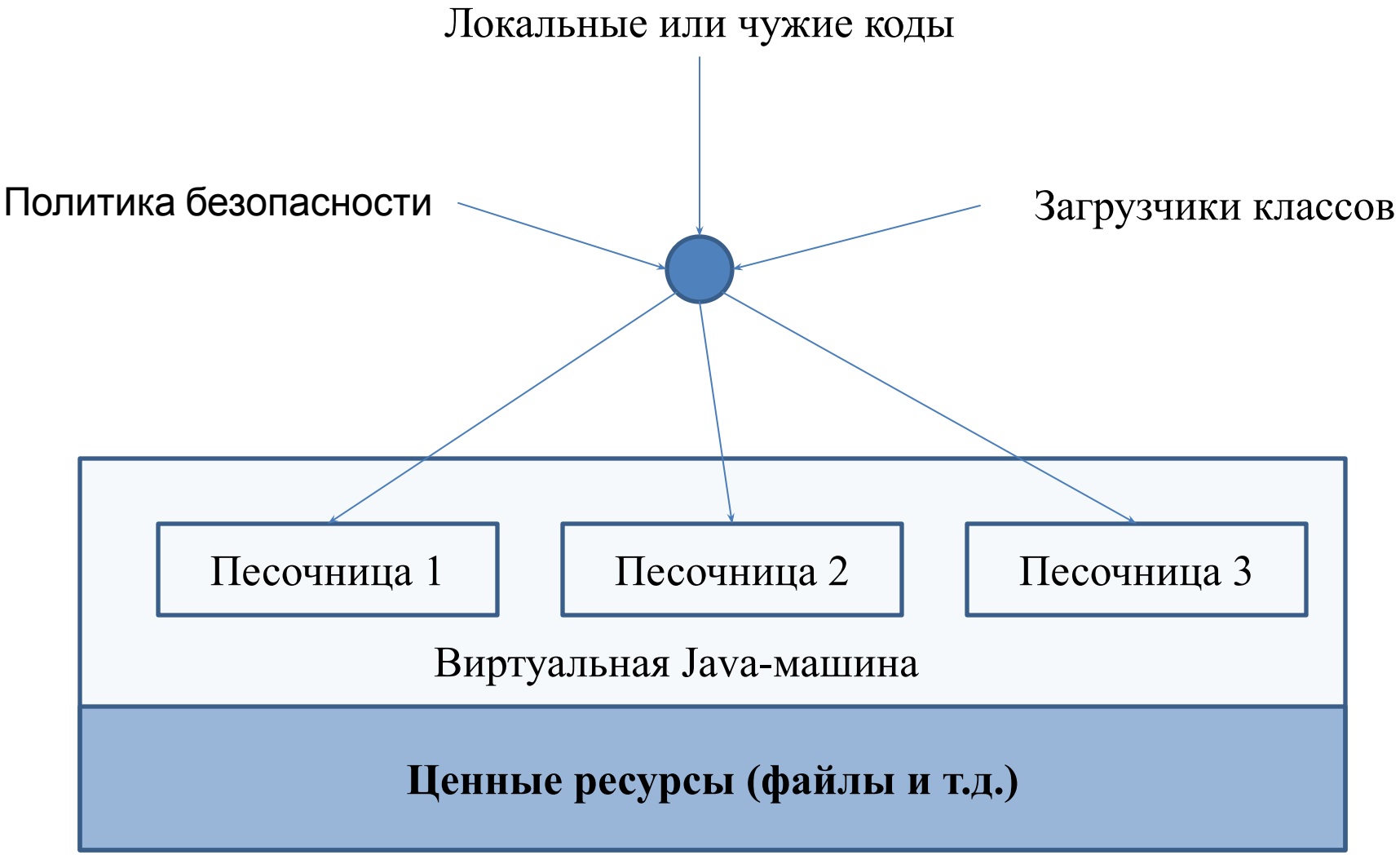
- Механизмы безопасности снабжены объектной оберткой

Организация защиты на платформе Java

Начиная с версии Java SE 1.2 на платформе Java предоставляется намного более гибкий механизм защиты. Правила защиты преобразуют источники кода в наборы полномочий, как показано на рисунке



Организация защиты на платформе Java



Организация защиты на платформе Java

Объектная организация механизмов безопасности

Механизмы безопасности оформлены в виде четырех основных пакетов и трех пакетов расширения:

- **java.security** — содержит интерфейсы и классы, составляющие каркас механизмов безопасности. Сюда входят рассмотренные выше средства разграничения доступа, а также криптографические средства, свободные от экспортного контроля США (выработка электронной подписи, вычисление хэш-функции и т.п.).
- **java.security.cert** — средства поддержки сертификатов X.509 версии 3.
- **java.security.interfaces** — средства генерации RSA- и DSA-ключей.
- **java.security.spec** — средства спецификации ключевого материала и параметров криптографических алгоритмов.

Организация защиты на платформе Java

Объектная организация механизмов безопасности

Нюанс состоит в том, что два последних элемента в списке сервисов (симметричное шифрование и выработка общего ключевого материала) подвержены экспортным ограничениям США, поэтому они, в отличие от остальных перечисленных сервисов, оформлены как расширение (Java Cryptography Extension, JCE), являющееся отдельным продуктом.

- **javax.crypto** — интерфейс и классы для симметричного шифрования (пакет расширения JCE 1.2).
- **javax.crypto.interfaces** — интерфейсы средств выработки ключей для алгоритма Диффи-Хелмана (пакет расширения JCE 1.2).
- **javax.crypto.spec** — классы для управления ключами и параметрами криптографических алгоритмов (пакет расширения JCE 1.2).