**Билет 12**

***Опишите пожалуйста, как работают, что общего и чем отличаются следующие потоки: InputStream, OutputStream, Reader, Writer? Дайте определение понятию «пул строк».***

Базовый класс InputStream представляет классы, которые получают данные из различных источников:

— массив байтов

— строка (String)

— файл

— канал (pipe): данные помещаются с одного конца и извлекаются с другого

— последовательность различных потоков, которые можно объединить в одном потоке

— другие источники (например, подключение к интернету)

Класс OutputStream — это абстрактный класс, определяющий потоковый байтовый вывод. В этой категории находятся классы, определяющие, куда направляются ваши данные: в массив байтов (но не напрямую в String; предполагается что вы сможете создать их из массива байтов), в файл или канал.

Символьные потоки имеют два основных абстрактных класса Reader и Writer, управляющие потоками символов Unicode. Класс Reader — абстрактный класс, определяющий символьный потоковый ввод. Класс Writer — абстрактный класс, определяющий символьный потоковый вывод. В случае ошибок все методы класса передают исключение IOException.

Пул строк (string pool) в Java, как понятно из описания, это некий пул (или список) объектов класса String, который хранится в специальном месте *кучи* (Java Heap). Разработчики Java сделали так, чтобы оптимизировать выделение памяти и хранение строк, ускорить и оптимизировать работу с ними.

Пул строк был создан по той простой причине, что строки - это самое используемое в программах на Java. Мы как минимум очень часто, а как максимум - почти всегда работает со строками в программах.

Пул строк работает следующим образом: когда мы создаем строку с помощью конструкции

String str = "HELLO";

Эта строка попадает в пул строк. Когда мы создаем другую строку с тем же значением:

String str2 = "HELLO";

То на самом деле сначала происходит поиск по пулу строк. И если там уже найдена такая строка - то str2 присваивается ссылка на уже созданный объект (на который указывает и str).

По этой причине следующая конструкция выведет true (вы же не забыли, что мы работаем не с самими объектами, а со ссылками на них?):

String str1 = "abc";

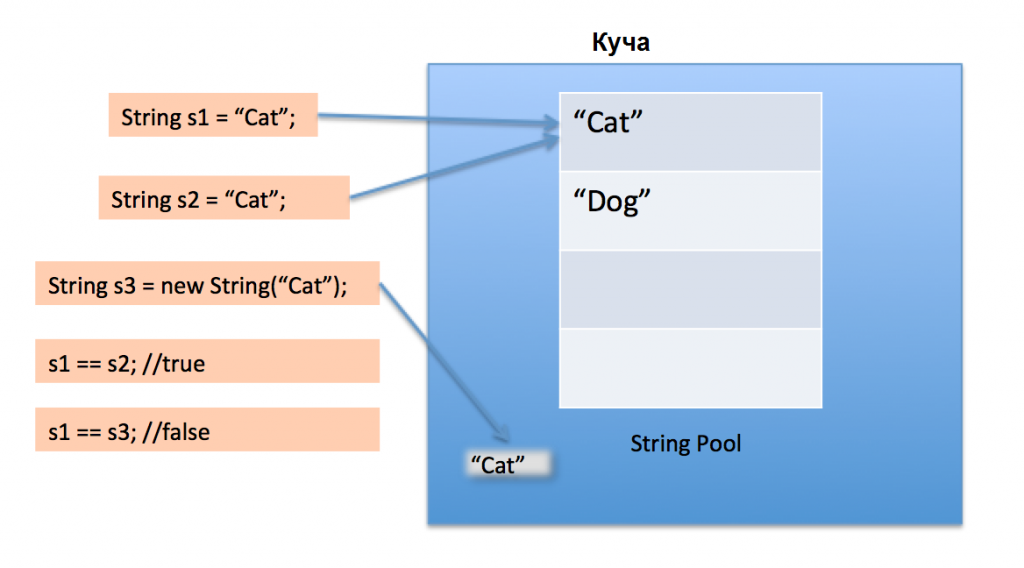
String str2 = "abc";

System.out.println(str1 == str2);

Тут может показаться, что мы создаем другой объект строки, но на самом деле это не так - происходит поиск по пулу строк.

Пул строк (**String Pool**) — это множество строк в кучи (**[Java Heap Memory](https://javadevblog.com/chto-takoe-heap-i-stack-pamyat-v-java.html" \t "_blank)**). Мы знаем, что String — особый класс в java, с помощью которого мы можем создавать строковые объекты.

На диаграмме ниже мы видим, как именно строковый пул расположен в памяти Java Heap. И как разные способы создания строк влияют на расположение их в памяти.

[](https://javadevblog.com/wp-content/uploads/2015/07/String-Pool-Java1.png)

Сам строковый пул возможен только потому, что [строки в Java неизменные](https://javadevblog.com/pochemu-stroki-v-java-neizmennye.html). Также пул строк позволяет сохранить память в Java Runtime, хотя это и требует больше времени на создание самой строки.

Ниже приведена программа, которая демонстрирует работу с пулом строк:

StringPool.java

package ua.com.prologistic.util;

public class StringPool {

public static void main(String[] args) {

String s1 = "Cat";

String s2 = "Cat";

String s3 = new String("Cat");

// сравниваем наши строки

System.out.println("s1 == s2 : " + ( s1 == s2 ) );

System.out.println("s1 == s3 : " + ( s1 == s3 ) );

}

}

Результат выполнения программы:

s1 == s2 : true

s1 == s3 : false