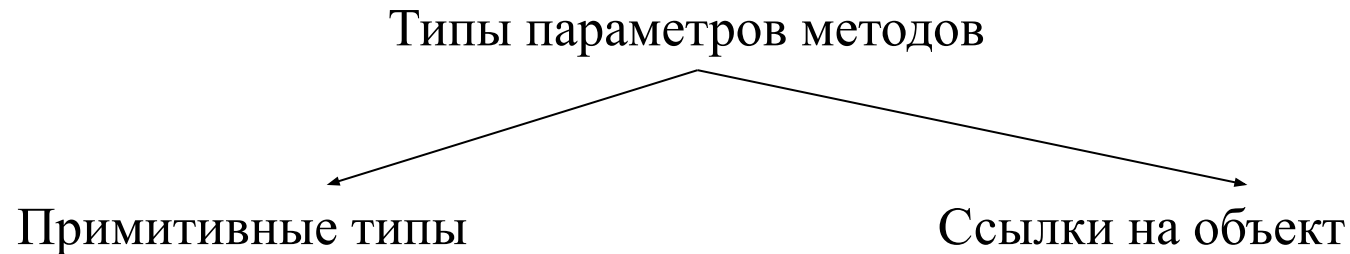


# Объектно-ориентированное программирование в JAVA

# Параметры методов

- Вызов по значению – метод получает значение, переданное ему из вызывающей части программы
- Вызов по ссылке – метод получает из вызывающей части программы местоположение переменной



## Метод:

- Не может изменять параметры примитивных типов
- Может изменять состояние объекта, передаваемого в качестве параметра
- Не может делать в своих параметрах ссылки на новые объекты

**Ссылка на материал для дополнительного изучения**

# Конструирование объектов

- Перегрузка – способ конструирования объектов, использующийся в том случае, если у нескольких методов имеются одинаковые имена, но разные параметры
  - Если значения поля не заданы явно, то происходит автоматическое присваивание значений по умолчанию:
    - Числа – нули
    - Логические переменные – значение false
    - Ссылки на объект – значение null
  - Если не определены конструкторы, то автоматически создается конструктор без аргументов. При этом всем полям экземпляра присваиваются их значения, предусмотренные по умолчанию
- ! Конструктор без аргументов вызывается только в том случае, если в классе не определены другие конструкторы

**Ссылка на материал для дополнительного изучения**

# Конструирование объектов

- **Константа** - публичное поле класса с модификатором *final*. Т.е. неизменяемое. Константы бывают уровня класса (*static*) и уровня объекта.

Константа уровня класса:

```
1. public class Parent
2. {
3.     public static final int MAX = 300;
4.     ...
5. }
```

Константа уровня объекта:

```
1. public class Parent
2. {
3.     public final int MAX = 400;
4.     ...
5. }
```

**Ссылка на материал для дополнительного изучения**

# Конструирование объектов

Имена параметров:

```
public Employee(String aName, double aSalary)
{
    name = aName;
    salary = aSalary;
}
```

**this** — ключевое слово, неявный параметр, то есть конструируемый объект

```
public Employee(String name, double salary)
{
    this.name = name;
    this.salary = salary;
}
```

**Ссылка на материал для дополнительного изучения**

# Конструирование объектов

- Блоки инициализации – выполняется всякий раз, когда создается объект данного класса
- Порядок инициализации:
  - инициализация полей в месте объявления и в инициализационном блоке происходит до инициализации в конструкторе
  - инициализации полей в месте объявления и в инициализационных блоках выполняются в порядке их объявления в классе
  - инициализация полей базового класса происходит полностью до инициализации производного класса, т.е. сначала выполняются все инициализаторы базового класса, а потом все инициализаторы производного класса.

```
// Статический блок инициализации
static
{
    var generator = new Random();
    nextId = generator.nextInt(10000);
}
```

**Ссылка на материал для дополнительного изучения**

# Конструирование объектов

- Деструктор – освобождает память, занятую объектами
- Ресурс должен быть освобожден сразу после его использования -> метод `close()`
- Освобождение ресурса ожидает до окончания срока действия виртуальной машины -> метод `Runtime.addShutdownHook()`

! Нельзя использовать метод `finalize()` для освобождения ненужных объектов из оперативной памяти

**Ссылка на материал для дополнительного изучения**

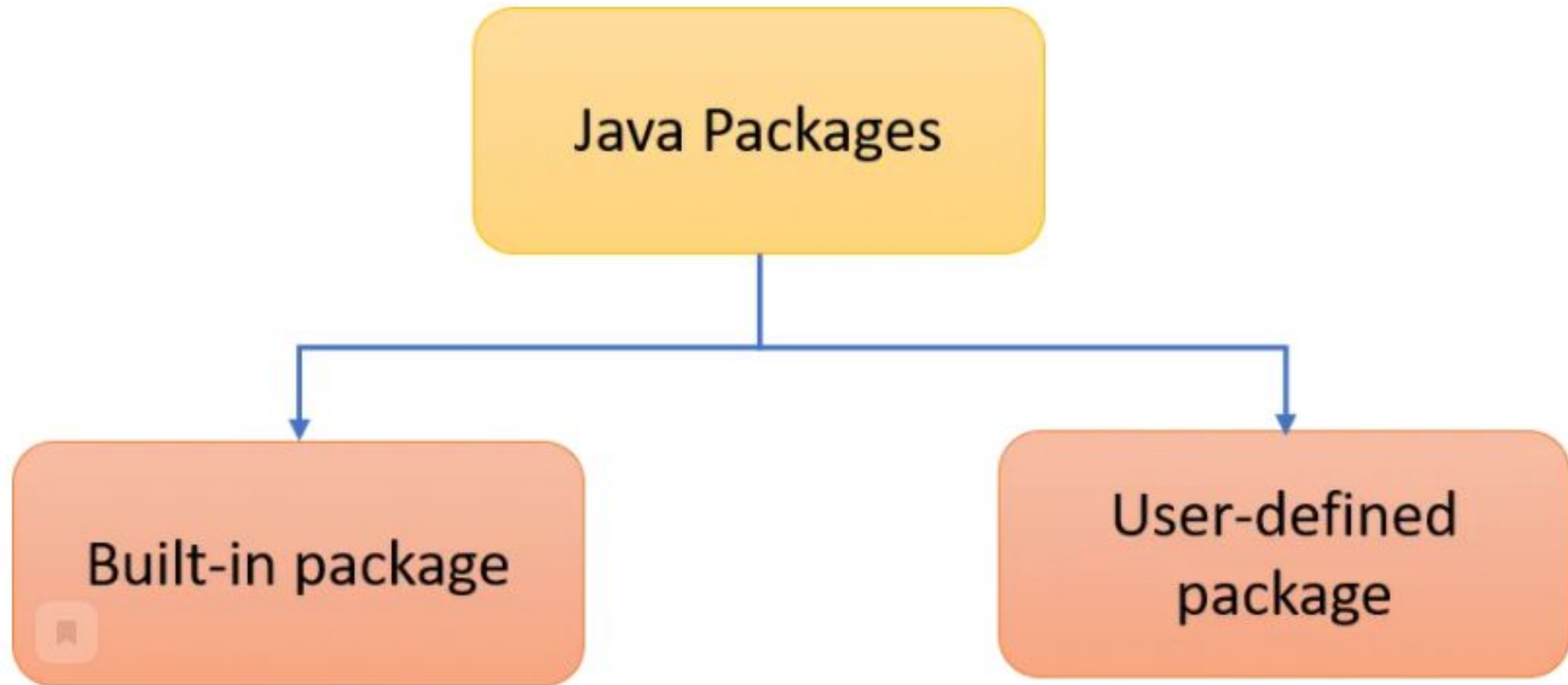
# Пакеты

- Пакеты в Java группируют несколько классов, интерфейсов или пакетов и т. д.
- Преимущества пакетов:
  - Избегает конфликтов именования классов. Это означает, что мы можем использовать одни и те же имена классов в двух разных пакетах
  - Обеспечивает возможность повторного использования путем доступа к классу из одного пакета в другом
  - Простота обслуживания, так как классы будут организованы
  - Обеспечивает защиту доступа для защищенных классов и классов по умолчанию
  - Это помогает в инкапсуляции или сокрытии данных

**Ссылка на материал для дополнительного изучения**



# Пакеты



# Пакеты

- Встроенный пакет – пакет, которые поставляются вместе с JDK
- `java.lang` - языковая поддержка
- `java.io` - операции ввода / вывода
- `java.util` - служебные классы для реализации структур данных
- `java.net` - сетевые операции
- `java.awt` - реализация графического пользовательского интерфейса
- `java.applet` - создание апплетов

# Пакеты

Импорт встроенного пакета:

```
//Imports the entire package
import packagename.*;

//Import specific class in package
import packagename.classname;
```

Пример:

```
import java.util.*;

public class SetArray {

    public static void main(String[] args) {
        Set<String> names = new HashSet<String>();
        names.add("Roshan");
        names.add("Kiran");
        names.add("Tejas");
        names.add("Karthik");

        String[] strnames = names.toArray(new String[names.size()]);
        for(String strvalues: strnames) {
            System.out.println(strvalues);
        }
        for(String strvalue:names) {
            System.out.println(strvalue);
        }

        System.out.println(new Date());

    }

}
```

**Ссылка на материал для дополнительного изучения**

# Пакеты

Импорт определенного класса в пакет:

```
//Import arraylist class of util package
import java.util.ArrayList;

public class ArrayListDemo2 {

    public static void main(String[] args) {
        //Create an Integer ArrayList
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        numbers.add(4);
        numbers.add(8);
        numbers.add(2);

        System.out.println("List elements: ");
        System.out.println(numbers);

        //Modify element
        numbers.set(1, 6);

        System.out.println("After modifying element at index 1:");
        System.out.println(numbers);

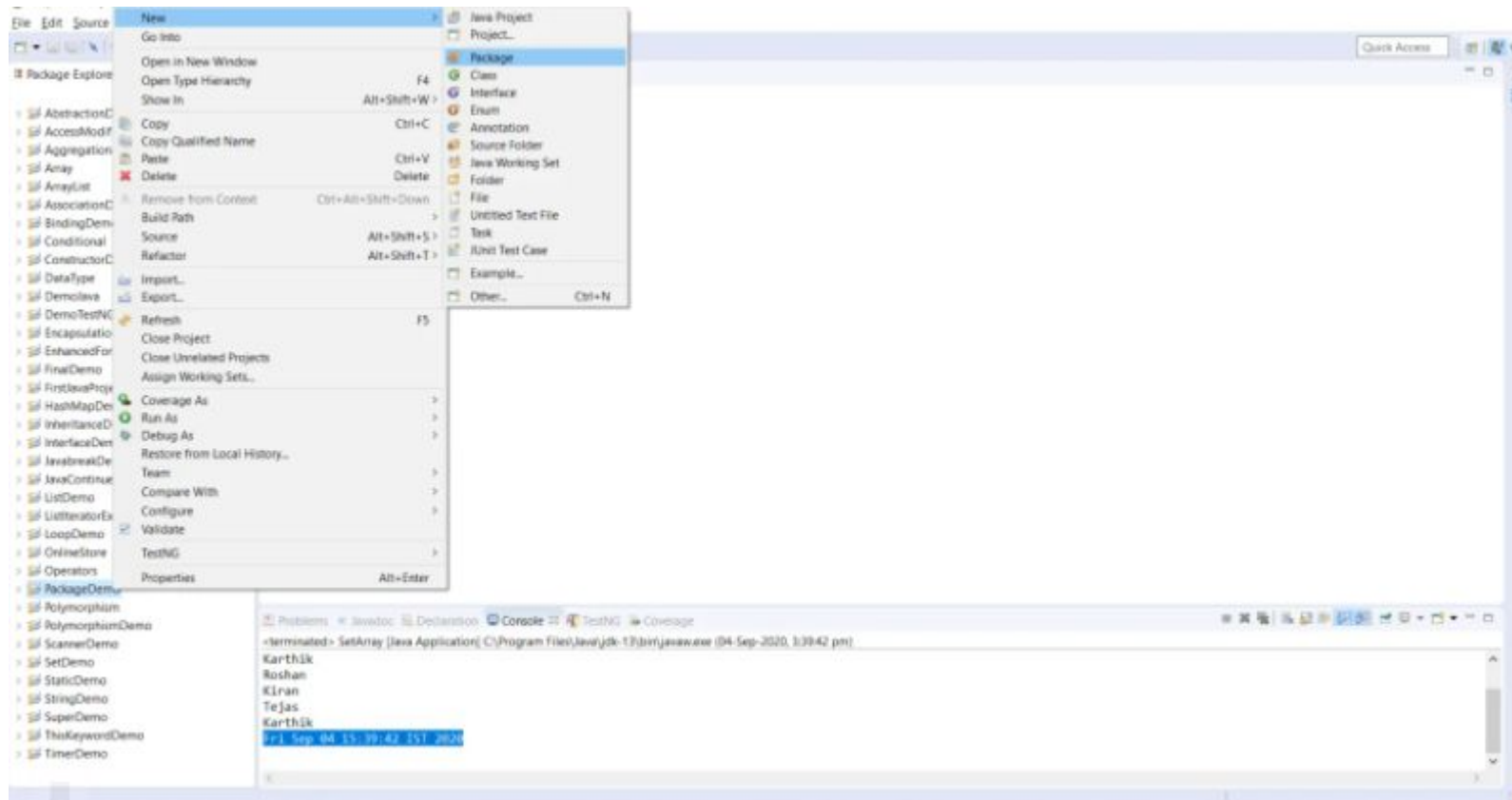
        //Remove an element
        numbers.remove(2);

        System.out.println("After removing element at index 2:");
        System.out.println(numbers);
    }
}
```


# Пакеты

- Пользовательский пакет – пакет, созданный вручную


Создание пакета:



# Пакеты

 New Java Package

**Java Package**  
Create a new Java package.




Creates folders corresponding to packages.

Source folder:

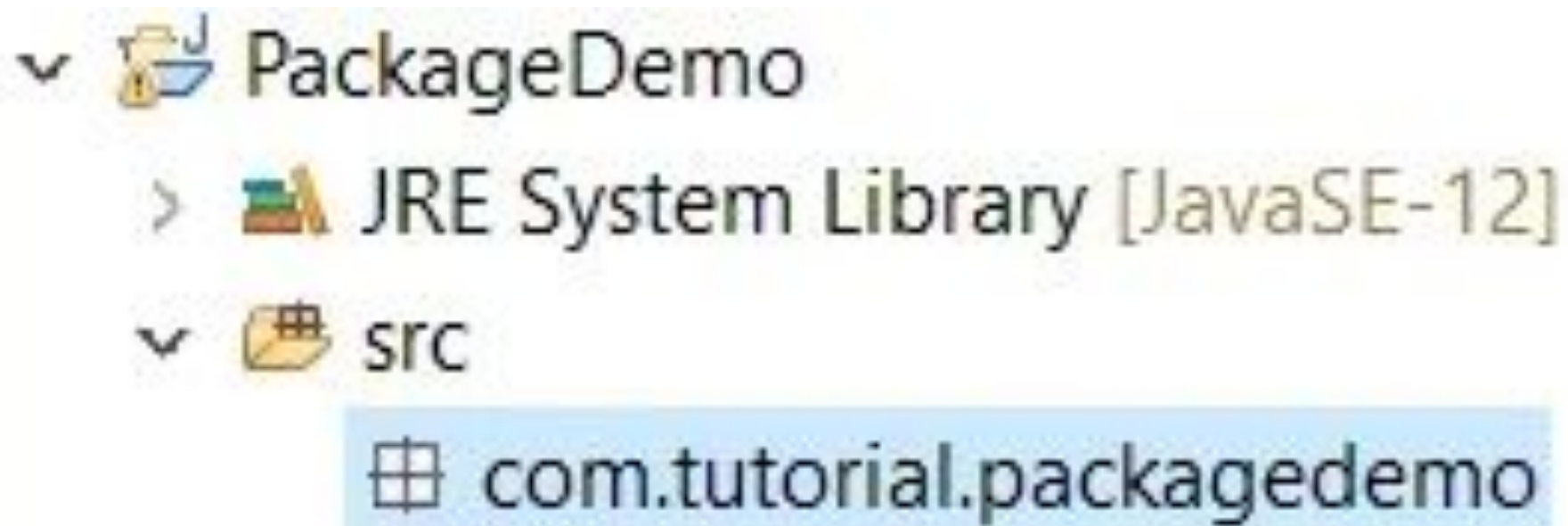
Name:

☐ Create package-info.java

☐ Generate comments (configure templates and default value [here](#))



# Пакеты





# Пакеты

Статический импорт:

```
import static java.lang.System.*;

public class StaticImportDemo {

    public static void main(String[] args) {
        out.println("Exmample of static import");
    }

}
```

**[Ссылка на материал для дополнительного изучения](#)**



- Создать абстрактный класс Car, и класс Engine

Класс Engine содержит поля - мощность, производитель.

Класс Car содержит поля - марка автомобиля, класс автомобиля, вес, мотор типа Engine. Абстрактные методы start(), stop(), и реализованные методы turnRight(), turnLeft()

Методы turnRight() и turnLeft() выводят на экран: "Поворот направо" или "Поворот налево".

Абстрактный метод printInfo(), который выводит полную информацию об автомобиле, и моторе.

Создать производный от Car класс - Lorry (грузовик), характеризующийся также грузоподъемностью кузов, реализовать методы start(), stop(). Метод start – выводит на экран «Грузовик поехал», метод stop - «грузовик остановился».

Создать производный от Car класс - SportCar, характеризующийся также предельной скоростью start(), stop(). Метод start – выводит на экран «SportCar поехал», метод stop - «SportCar остановился».