Java Crash Course

Java Crash Course: Main Class

```java
package tutorial;

public class JavaExample {

  public static void main(String[] args) {
    System.out.println("Hello MATSim!");
  }

}
```

Java Crash Course: Class

```java
public class Person {
    private String name;

    // Constructor, create a new object and initialize it
    public Person(String name) {
        this.name = name;
    }

    // getter
    public String getName() {
        return this.name;
    }

    // setter
    public void setName(String name) {
        this.name = name;
    }
}
```

```java
Person p1 = new Person("Alice");
System.out.println(p1.getName()); // prints out "Alice"
p1.setName("Bob");
System.out.println(p1.getName()); // prints out "Bob"
```

A class is like a template, and instance of it is called an object.

Java Crash Course: List

```
List<Person> persons = new ArrayList<>();

Person person1 = new Person("Alice");
Person person2 = new Person("Bob");

persons.add(person1);
persons.add(person2);

System.out.println("# persons: " + persons.size());

for (Person p : persons) {
  System.out.println(p.getName());
}
```

Java Crash Course: Map (Lookup Table)

```
Map<Person, Integer> ageMap = new HashMap<>();

Person person1 = new Person("Alice");
Person person2 = new Person("Bob");

ageMap.put(person1, 45);
ageMap.put(person2, 42);

System.out.println("# entries: " + ageMap.size());

System.out.println("age of Alice: " + ageMap.get(person1));

for (Person p : ageMap.keySet()) {
  System.out.println(p.getName());
}

for (Integer age : ageMap.values()) {
  System.out.println(age);
}

for (Map.Entry<Person, Integer> entry : ageMap.entrySet()) {
  Person p = entry.getKey();
  Integer age = entry.getValue();
  System.out.println(p.getName() + ": " + age);
}
```

Java Crash Course: Arrays

```
String[] names = new String[3];

names[0] = "Alice";
names[1] = "Bob";
names[2] = "Carol";

System.out.println("# entries: " + names.length);

for (String name : names) {
  System.out.println(name);
}
```

Java Crash Course: Types

```
// primitive types

int     i1 = 0;    // integer number, 32 bits, -2^31 ... +(2^31 - 1)
long    l1 = 0;    // long integer, 64 bits, -2^63 ... +(2^63 - 1)
float   f1 = 0.0f; // floating-point number, 32 bits, 6-7 digits of precision
double  d1 = 0.0;  // floating-point number, 64 bits, 15-16 digits of precision
boolean b1 = true; // true or false
char    c1 = 'M';  // single character

// object types

String  s1 = "Hello MATSim"; // text, characters

Integer i2 = 0;    // integer number as object, use in Lists and Maps
Long    l2 = 0;    // long integer number as object, use in Lists and Maps
Float   f2 = 0;    // floating-point number as object, use in Lists and Maps
Double  d2 = 0;    // floating-point number as object, use in Lists and Maps
Boolean b2 = true; // boolean as object, use in Lists and Maps
```

Java Crash Course: Conditions

```
if (i == 5) {
  System.out.println("i is equal to 5.")
}
if (i < 3 || i > 6) { // 'or'
  System.out.println("i is smaller than 3 or larger than 6.");
}
if (i >=3 && i <= 6) { // 'and'
  System.out.println("i is between 3 and 6.");
}
```

```
if (s.equals("one")) {
  System.out.println("The string s is 'one'.")
}
if (s.startsWith("two")) {
  System.out.println("The string s starts with 'two'.")
}
if (s.endsWith("three")) {
  System.out.println("The string s ends with 'three'.")
}
```

Java Crash Course: Objects

Objects (instances of classes) have an inner state.
Multiple variables can point to the same objects.

```
Person p1 = new Person("Alice");
System.out.println(p1.getName()); // prints out "Alice"

Person p2 = p1; // this does *not* make a copy! p1 and p2 refer to the same object now.

System.out.println(p2.getName()); // prints out "Alice"

p2.setName("Bob");
System.out.println(p2.getName()); // prints out "Bob"

System.out.println(p1.getName()); // prints out "Bob" too!
```

— Java Crash Course —

Maven

Managing Dependencies

If you write code, how does Java know you want to use MATSim?

Java packages code into JAR files (java archive).
An application usually consists of multiple jar files.
Downloading and managing all the different jar files, all with the correct version, is difficult.

Maven (mvn) manages all the jar files an application requires → **Dependency Management**

Maven also helps to package all code and dependencies together for a release.

Maven

Each Maven project has a `pom.xml` that describes characteristics and dependencies of a project.

```
<project xmlns="..." ... >
  <groupId>org.matsim</groupId>
  <artifactId>matsim-example-project</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <name>MATSim example project</name>
  <description>MATSim example project</description>

  <repositories>
    ...
  </repositories>

  <dependencies>
    <dependency>
      <!-- This is the matsim "core": -->
      <groupId>org.matsim</groupId>
      <artifactId>matsim</artifactId>
      <version>11.0</version>
    </dependency>

    ...
  </dependencies>

  ...
</project>
```

Maven

To use MATSim extensions, add the corresponding dependencies to the pom.xml:

```
...
<dependencies>
  <dependency>
    <groupId>org.matsim</groupId>
    <artifactId>matsim</artifactId>
    <version>11.0</version>
  </dependency>
  ...

  <dependency>
    <groupId>org.matsim.contrib</groupId>
    <artifactId>drt</artifactId>
    <version>11.0</version>
  </dependency>

  <dependency>
    <groupId>org.matsim.contrib</groupId>
    <artifactId>dvrp</artifactId>
    <version>11.0</version>
  </dependency>
</dependencies>
```

Make sure that versions match.