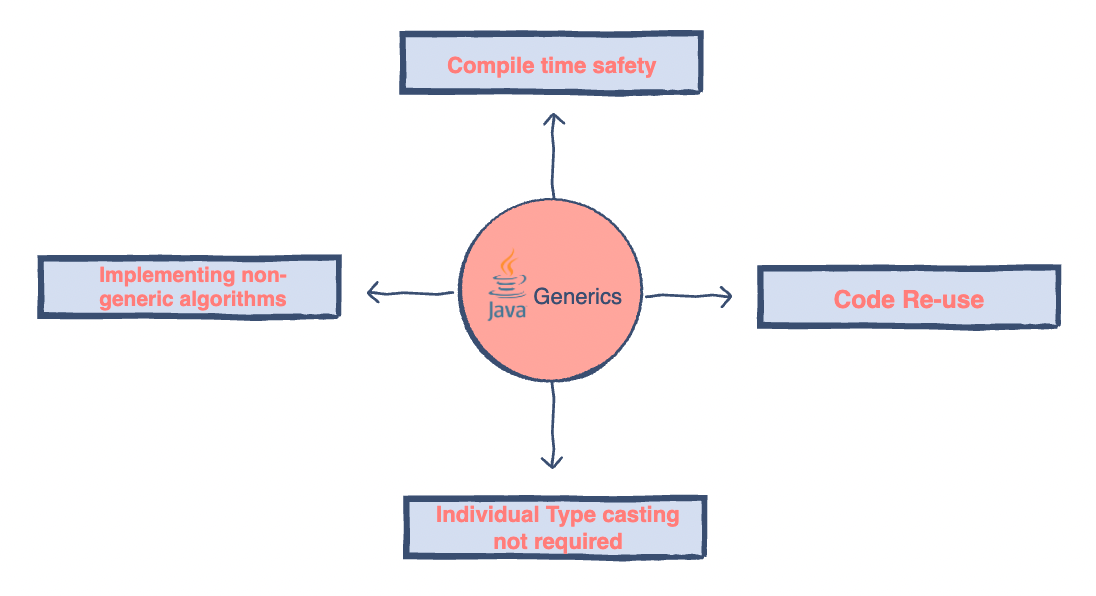
**Java Generics** is a **set of related methods** or a **set of similar types**. **Generics allow types Integer, String, or even user-defined types to be passed as a parameter to classes, methods, or interfaces**. Generics are mostly used by classes like HashSet or HashMap.

**Advantages of using generics**

* Generics ensure **compile-time safety** which allows the programmer to catch the invalid types while compiling the code.
* Java Generics helps the programmer to **reuse the code** for whatever type he/she wishes. For instance, a programmer writes a generic method for sorting an array of objects. Generics allow the programmer to use the same method for Integer arrays, Double arrays, and even String arrays.
* Another advantage of using generics is that **Individual typecasting isn’t required**. The programmer defines the initial type and then lets the code do its job.
* It allows us to **implement non-generic algorithms**.

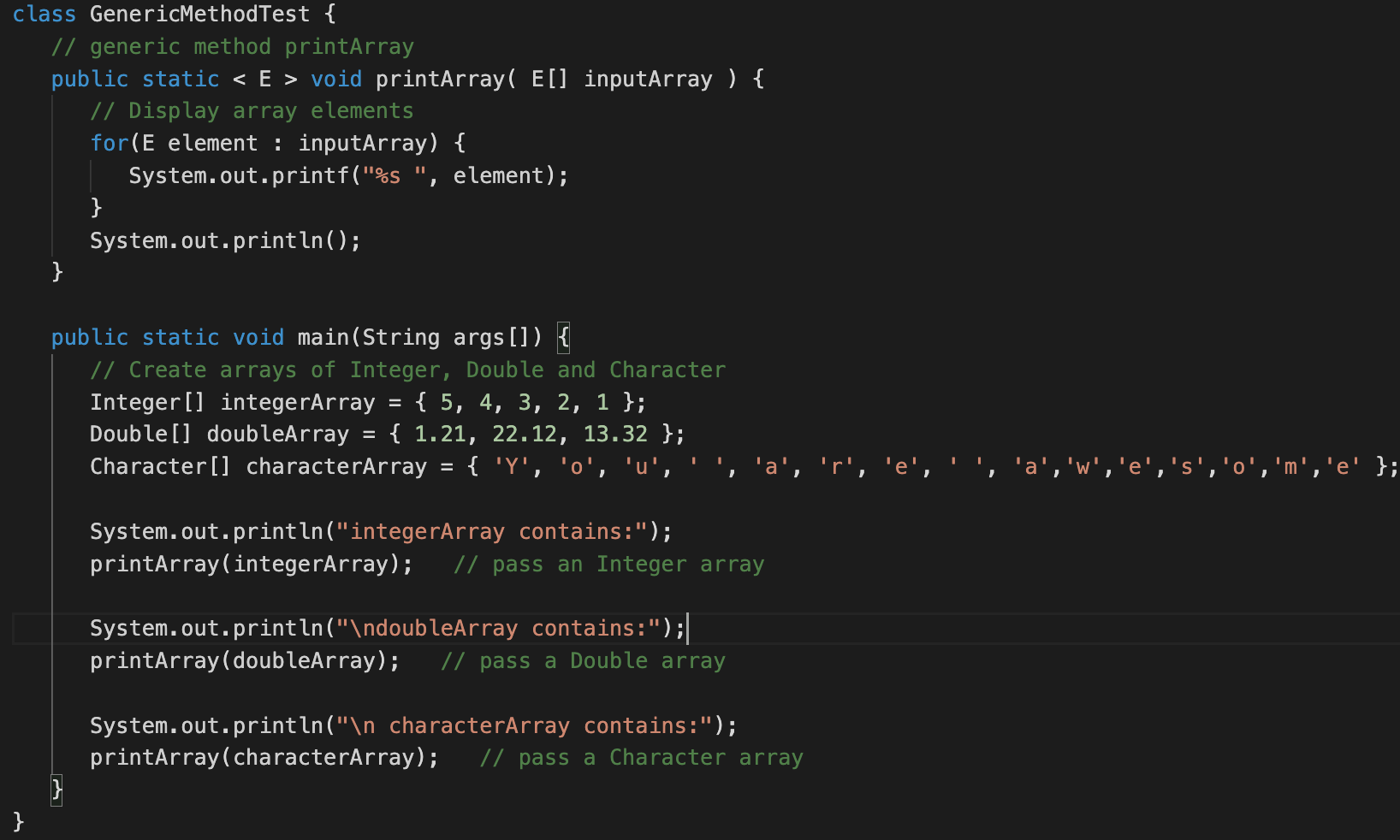


**Constraints**

* Java Generics **does not support sub-typing**.
* We **cannot create generic arrays** in Java Generics.

## **Types of Java Generics**

**Generic method**: **Generic Java method takes a parameter and returns some value after performing a task**. It is exactly like a normal function, however, a generic method has type parameters that are cited by actual type. This allows the generic method to be used in a more general way. The **compiler takes care of the type of safety** which enables programmers to code easily since they do not have to perform long, individual type castings.



**Generic classes**: A **generic class is implemented exactly like a non-generic class**. The **only difference is that it** **contains a type parameter section**. There can be **more than one type of parameter, separated by a comma**. **The classes, which accept one or more parameters, ​are known as parameterized classes or parameterized types**.

****