Дженерики

В ряде задач внутри класса могут использоваться ссылки на объекты разных классов. Причем в разных обстоятельствах используемые классы могут быть разными. Допустим, что в одной части задачи нам необходимо иметь ссылку на объект класса **String**, а в другой — **Integer**. Без использования дженериков единственное решение — хранить объект в виде переменной общего родительского класса. Самым универсальным решением является использование типа **Object** для переменной. В этом случае переменная может хранить ссылку на объекты любого класса, так как **Object** является предком всех классов.

Но если вы храните объект в переменной **Object**, то при его получении возникает проблема приведения типов и определения типа объекта. Если вы положили объект одного типа, а пытаетесь извлечь другого, то у вас будут ошибки времени выполнения. Например, у нас имеется класс:

```
public class Box {
	private Object item;
	public Object getItem() {
		return item;
	}
	public void setItem(Object item){
		this.item = item;
	}
}
```

Если мы попытаемся им воспользоваться таким образом:

```
Box box = new Box();
box.setItem("Test");
Object item = box.getItem();
Integer itemInt = (Integer)item;
```

Во время компиляции никаких ошибок возникать не будет, а во время выполнения вы получите ошибку выполнения ClassCastException, так как в объекте хранится ***String***, а мы пытаемся его занести в переменную ***Integer***.

Решить подобную проблему можно, изменив класс **Box** следующим образом:

```
public class Box<T> {
	private T item;
	public T getItem() {
		return item;
	}
```

```
        public void setItem(T item){
                this.item = item;
        }
}
```

Это и называется **дженериком**. При использовании этого класса его объект надо создавать с указанием в угловых скобках класса хранимого элемента. Причем делается это обычно два раза: при объявлении переменной и при создании объекта с помощью new.

```
Box<String> myBox = new Box<>();

myBox.setItem("Test");

String s = myBox.getItem();
```

Чем эта ситуация отличается от предыдущей? Во-первых, вам не позволят сделать присвоение переменной неправильного типа. Если вы здесь напишете Integer s вместо String s, ошибку вы получите еще на стадии компиляции. Во-вторых, сразу возвращается значение правильного типа и вам не требуется лишний раз делать приведение типов в вашей программе.