Enum Java

Un enum di Java è uno speciale tipo di “classe” che contiene una lista di costanti. Per dichiarare un enum è necessario usare la parola chiave **enum** al posto di class o interface.

```
public enum Giorni {
    LUNEDI,
    MARTEDI,
    MERCOLEDI,
    GIOVEDI,
    VENERDI,
    SABATO,
    DOMENICA;
}
```

Questo è un esempio di enum che contiene solo i giorni della settimana.

Per accedere a un valore di un enum occorre solo usare la dot notation (come con tutte le classi), come se i valori all’interno fossero attributi statici.

```
Giorni giorno = Giorni.LUNEDI;
```

Grazie agli enum, quindi, possiamo avere dei nuovi tipi di dato che possono assumere solo i valori che gli specifichiamo.
È utile usare gli enum quando la scelta che può effettuare l’utente è vincolata a particolari valori, ad esempio un giorno della settimana, un ruolo nella squadra di calcio oppure un livello che può essere solo basso, medio o alto.
Nota: le variabili all’interno dell’enum sono tutte **public**, **static** e **final**. Un enum non può essere istanziato per creare oggetti **oltre a quelli specificati.**

In ogni enum ci sono una serie di metodi utili per lavorarci, come ad esempio:

Enum.values( )	Ritorna un array del tipo dell’enum contenente tutte le costanti.	{Giorni.LUNEDI, Giorni.MARTEDI, …, Giorni.DOMENICA}
Enum.valueOf(String name)	Passandogli una stringa, ritorna la costante corrispondente. Lancia un’eccezione se non trova alcuna costante ed è case sensitive.	Giorni.valueOf(“DOMENICA”) funziona ma Giorni.valueOf(“domenica”) no

Inoltre ogni costante ha i suoi metodi, tra quelli di default ci sono:

.name( )	Ritorna il nome con cui è stata dichiarata la costante. Non è possibile farne l’override.	giorno.name( ) ritorna “LUNEDI” ad esempio
.ordinal( )	Ritorna la posizione della costante nella lista partendo da 0. Non è possibile farne l’override.	giorno.ordinal( ) ritorna 0 per lunedì, 1 per martedì e così via
.equals(Object other)	Ritorna true o false. Non è possibile farne l’override.	giorno.equals(Giorni.DOMENICA) ritorna false
.toString( )	Un normale toString, di questo si può fare l’override. Come si fa?	

Ogni costante può avere dei propri attributi e dei propri metodi. Se volessimo, ad esempio, associare ad ogni giorno una stringa contenente il nome ed una contenente l'abbreviazione potremmo fare in questo modo:

```java
public enum Giorni {
    LUNEDI( nome: "Lunedì",  abbreviazione: "Lun"),
    MARTEDI( nome: "Martedì",  abbreviazione: "Mar"),
    MERCOLEDI( nome: "Mercoledì",  abbreviazione: "Mer"),
    GIOVEDI( nome: "Giovedì",  abbreviazione: "Gio"),
    VENERDI( nome: "Venerdì",  abbreviazione: "Ven"),
    SABATO( nome: "Sabato",  abbreviazione: "Sab"),
    DOMENICA( nome: "Domenica",  abbreviazione: "Dom");

    private final String nome;
    private final String abbreviazione;

    Giorni(String nome, String abbreviazione) {
        this.nome = nome;
        this.abbreviazione = abbreviazione;
    }

    public String getNome() {
        return this.nome;
    }

    public String getAbbreviazione() {
        return this.abbreviazione;
    }

    @Override
    public String toString() {
        return "Giorni{" +
                "nome='" + nome + '\'' +
                ", abbreviazione='" + abbreviazione + '\'' +
                '}';
    }
}
```

1.Mettere le variabili (nome ed abbreviazione)

2.Creare il costruttore con nome ed abbreviazione come parametri

3.Modificare le costanti sopra specificate mettendo le parentesi tonde e tutti i parametri necessari

Poi si possono creare anche metodi che usano quelle variabili, per accedere a questi metodi è necessaria la dot notation.

```java
System.out.println(giorno.getAbbreviazione());
```