Any

Any é pro **Kotlin** o que **Object é pro java**, a classe pai de todas as outras classes. Então ***Equals e Hashcode*** no **Kotlin** são feitos a partir do **Any**. Objetos do tipo **Any também podem devolver seu tipo através do "cast":**

```
fun converte (funcionarioAny : Any){
    if(funcionarioAny is Funcionario){
        total = funcionarioAny.bonificacao;
    }

}
```

Equals também é proveniente do **Any, equals** é a **comparação entre objetos pra saber se as instâncias representam a mesma referência de objeto**. **Hashcode** é o **código "real" do objeto**, sempre que a referência for chamada ela terá seu **hashcode** *(tipo um RG)*.

O lance desses **métodos** *(equals, hashcode e toString)* de **Any** é que eles **podem *(muitas vezes devem)* ser sobrescritos nas classes específicas,** isso porque o ***Equals e Hashcode*** usam por **padrão a referência do objeto** e não as informações que **ele** representa:

```
override fun toString(): String {
    return "Endereco(logradouro='$logradouro', numero=$numero, bairro='$bairro', cidade='$cidade'," +
            " estado='$estado', cep='$cep', complemento='$complemento')"
}

override fun equals(other: Any?): Boolean {
    if (this === other) return true
    if (javaClass != other?.javaClass) return false

    other as Endereco

    if (logradouro != other.logradouro) return false
    if (numero != other.numero) return false
    if (cep != other.cep) return false

    return true
}

override fun hashCode(): Int {
    var result = logradouro.hashCode()
    result = 31 * result + numero
    result = 31 * result + cep.hashCode()
    return result
}
```