SCIENCES SORBONNE UNIVERSITÉ

SRCS : Systèmes Répartis Client/Serveur

TD 2 : Exploiter le code au moment de l'exécution

Objectifs pédagogiques :

- **Réflexivité**
- **Annotation**
- **Polymorphisme**

Exercice 1 – Un `toString` pour tous

Écrire une méthode statique `String toString(Object o)` générique (au sens général, pas au sens généricité Java) qui permet d'avoir une représentation d'un objet quelconque sous la forme d'une chaîne caractère. On doit respecter le format suivant

```
NomDeClasseReelle(typeatt1 name_att1=val_att1, typeatt2 name_att2=val_att2, ....)
```

On ne prendra pas en compte les attributs statiques.

Exercice 2 – Connaître ses types

Écrire une méthode statique `getTypes` qui pour un objet `o` donné renvoie la liste des types de cet objet.

Exercice 3 – Annotation et introspection

Nous souhaitons écrire une annotation `ExecuteOnBuilding` de méthode qui indique si une méthode doit être exécutée automatiquement après l'instanciation de l'objet. On pourra paramétrer l'annotation par un entier qui indiquera le nombre de fois que l'on souhaite exécuter la méthode (valeur par défaut = 1). On supposera que les méthodes ne prennent pas d'argument et ne retourne rien.

Question 1

Donner le code de l'annotation `ExecuteOnBuilding`

Question 2

Écrire une méthode statique `deploy` qui prend une classe en paramètre, qui instancie une nouvelle instance de cette classe, qui exécute les méthodes marquées par l'annotation `ExecuteOnBuilding` et qui renvoie un objet du type de la classe.

Class Class<T>

- java.lang.Object
  - java.lang.Class<T>

- Type Parameters:
  `T` - the type of the class modeled by this `Class` object. For example, the type of `String.class` is `Class<String>`. Use `Class<?>` if the class being modeled is unknown.

  All Implemented Interfaces:
  Serializable, AnnotatedElement, GenericDeclaration, Type

```
public final class Class<T>
extends Object
implements Serializable, GenericDeclaration, Type, AnnotatedElement
```

Instances of the class Class represent classes and interfaces in a running Java application. An enum is a kind of class and an annotation is a kind of interface. Every array also belongs to a class that is reflected as a Class object that is shared by all arrays with the same element type and number of dimensions. The primitive Java types (boolean, byte, char, short, int, long, float, and double), and the keyword void are also represented as Class objects. Class has no public constructor. Instead Class objects are constructed automatically by the Java Virtual Machine as classes are loaded and by calls to the defineClass method in the class loader.

The following example uses a `Class` object to print the class name of an object:

```
void printClassName(Object obj) {
    System.out.println("The class of " + obj +
                       " is " + obj.getClass().getName());
}
```

Modifier and Type	Method and Description
`<U> Class<? extends U>`	`asSubclass(Class<U> clazz)` Casts this `Class` object to represent a subclass of the class represented by the specified class object.
`T`	`cast(Object obj)` Casts an object to the class or interface represented by this `Class` object.
`boolean`	`desiredAssertionStatus()` Returns the assertion status that would be assigned to this class if it were to be initialized at the time this method is invoked.
`static Class<?>`	`forName(String className)` Returns the `Class` object associated with the class or interface with the given string name.
`static Class<?>`	`forName(String name, boolean initialize, ClassLoader loader)` Returns the `Class` object associated with the class or interface with the given string name, using the given class loader.
`AnnotatedType[]`	`getAnnotatedInterfaces()` Returns an array of `AnnotatedType` objects that represent the use of types to specify superinterfaces of the entity represented by this `Class` object.
`AnnotatedType`	`getAnnotatedSuperclass()` Returns an `AnnotatedType` object that represents the use of a type to specify the superclass of the entity represented by this `Class` object.
`<A extends Annotation> A`	`getAnnotation(Class<A> annotationClass)` Returns this element's annotation for the specified type if such an annotation is *present*, else null.
`Annotation[]`	`getAnnotations()` Returns annotations that are *present* on this element.
`<A extends Annotation> A[]`	`getAnnotationsByType(Class<A> annotationClass)` Returns annotations that are *associated* with this element.
`String`	`getCanonicalName()` Returns the canonical name of the underlying class as defined by the Java Language Specification.
`Class<?>[]`	`getClasses()` Returns an array containing `Class` objects representing all the public classes and interfaces that are members of the class represented by this `Class` object.
`ClassLoader`	`getClassLoader()` Returns the class loader for the class.
`Class<?>`	`getComponentType()` Returns the `Class` representing the component type of an array.
`Constructor<T>`	`getConstructor(Class<?>... parameterTypes)` Returns a `Constructor` object that reflects the specified public constructor of the class represented by this `Class` object.
`Constructor<?>[]`	`getConstructors()` Returns an array containing `Constructor` objects reflecting all the public constructors of the class represented by this `Class` object.
`<A extends Annotation> A`	`getDeclaredAnnotation(Class<A> annotationClass)` Returns this element's annotation for the specified type if such an annotation is *directly present*, else null.
`Annotation[]`	`getDeclaredAnnotations()` Returns annotations that are *directly present* on this element.
`<A extends Annotation> A[]`	`getDeclaredAnnotationsByType(Class<A> annotationClass)` Returns this element's annotation(s) for the specified type if such annotations are either *directly present* or *indirectly present*.
`Class<?>[]`	`getDeclaredClasses()` Returns an array of `Class` objects reflecting all the classes and interfaces declared as members of the class represented by this `Class` object.
`Constructor<T>`	`getDeclaredConstructor(Class<?>... parameterTypes)` Returns a `Constructor` object that reflects the specified constructor of the class or interface represented by this `Class` object.
`Constructor<?>[]`	`getDeclaredConstructors()` Returns an array of `Constructor` objects reflecting all the constructors declared by the class represented by this `Class` object.
`Field`	`getDeclaredField(String name)` Returns a `Field` object that reflects the specified declared field of the class or interface represented by this `Class` object.
`Field[]`	`getDeclaredFields()` Returns an array of `Field` objects reflecting all the fields declared by the class or interface represented by this `Class` object.
`Method`	`getDeclaredMethod(String name, Class<?>... parameterTypes)` Returns a `Method` object that reflects the specified declared method of the class or interface represented by this `Class` object.
`Method[]`	`getDeclaredMethods()` Returns an array containing `Method` objects reflecting all the declared methods of the class or interface represented by this `Class` object, including public, protected, default (package) access, and private methods, but excluding inherited methods.
`Class<?>`	`getDeclaringClass()` If the class or interface represented by this `Class` object is a member of another class, returns the `Class` object representing the class in which it was declared.
`Class<?>`	`getEnclosingClass()` Returns the immediately enclosing class of the underlying class.
`Constructor<?>`	`getEnclosingConstructor()` If this `Class` object represents a local or anonymous class within a constructor, returns a `Constructor` object representing the immediately enclosing constructor of the underlying class.
`Method`	`getEnclosingMethod()` If this `Class` object represents a local or anonymous class within a method, returns a `Method` object representing the immediately enclosing method of the underlying class.
`T[]`	`getEnumConstants()` Returns the elements of this enum class or null if this Class object does not represent an enum type.
`Field`	`getField(String name)` Returns a `Field` object that reflects the specified public member field of the class or interface

	represented by this `Class` object.
`Field[]`	`getFields()` Returns an array containing `Field` objects reflecting all the accessible public fields of the class or interface represented by this `Class` object.
`Type[]`	`getGenericInterfaces()` Returns the `Type`s representing the interfaces directly implemented by the class or interface represented by this object.
`Type`	`getGenericSuperclass()` Returns the `Type` representing the direct superclass of the entity (class, interface, primitive type or void) represented by this `Class`.
`Class<?>[]`	`getInterfaces()` Determines the interfaces implemented by the class or interface represented by this object.
`Method`	`getMethod(String name, Class<?>... parameterTypes)` Returns a `Method` object that reflects the specified public member method of the class or interface represented by this `Class` object.
`Method[]`	`getMethods()` Returns an array containing `Method` objects reflecting all the public methods of the class or interface represented by this `Class` object, including those declared by the class or interface and those inherited from superclasses and superinterfaces.
`int`	`getModifiers()` Returns the Java language modifiers for this class or interface, encoded in an integer.
`String`	`getName()` Returns the name of the entity (class, interface, array class, primitive type, or void) represented by this `Class` object, as a `String`.
`Package`	`getPackage()` Gets the package for this class.
`ProtectionDomain`	`getProtectionDomain()` Returns the `ProtectionDomain` of this class.
`URL`	`getResource(String name)` Finds a resource with a given name.
`InputStream`	`getResourceAsStream(String name)` Finds a resource with a given name.
`Object[]`	`getSigners()` Gets the signers of this class.
`String`	`getSimpleName()` Returns the simple name of the underlying class as given in the source code.
`Class<? super T>`	`getSuperclass()` Returns the `Class` representing the superclass of the entity (class, interface, primitive type or void) represented by this `Class`.
`String`	`getTypeName()` Return an informative string for the name of this type.
`TypeVariable<Class<T>>[]`	`getTypeParameters()` Returns an array of `TypeVariable` objects that represent the type variables declared by the generic declaration represented by this `GenericDeclaration` object, in declaration order.
`boolean`	`isAnnotation()` Returns true if this `Class` object represents an annotation type.
`boolean`	`isAnnotationPresent(Class<? extends Annotation> annotationClass)` Returns true if an annotation for the specified type is *present* on this element, else false.
`boolean`	`isAnonymousClass()` Returns `true` if and only if the underlying class is an anonymous class.
`boolean`	`isArray()` Determines if this `Class` object represents an array class.
`boolean`	`isAssignableFrom(Class<?> cls)` Determines if the class or interface represented by this `Class` object is either the same as, or is a superclass or superinterface of, the class or interface represented by the specified `Class` parameter.
`boolean`	`isEnum()` Returns true if and only if this class was declared as an enum in the source code.
`boolean`	`isInstance(Object obj)` Determines if the specified `Object` is assignment-compatible with the object represented by this `Class`.
`boolean`	`isInterface()` Determines if the specified `Class` object represents an interface type.
`boolean`	`isLocalClass()` Returns `true` if and only if the underlying class is a local class.
`boolean`	`isMemberClass()` Returns `true` if and only if the underlying class is a member class.
`boolean`	`isPrimitive()` Determines if the specified `Class` object represents a primitive type.
`boolean`	`isSynthetic()` Returns `true` if this class is a synthetic class; returns `false` otherwise.
`T`	`newInstance()` Creates a new instance of the class represented by this `Class` object.
`String`	`toGenericString()` Returns a string describing this `Class`, including information about modifiers and type parameters.
`String`	`toString()` Converts the object to a string.

- **Methods inherited from class java.lang.Object**

  `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

java.lang.reflect

Class Field

- java.lang.Object
  - java.lang.reflect.AccessibleObject
    - java.lang.reflect.Field
- All Implemented Interfaces:
  AnnotatedElement, Member

```
public final class Field
extends AccessibleObject
implements Member
```

A `Field` provides information about, and dynamic access to, a single field of a class or an interface. The reflected field may be a class (static) field or an instance field. A `Field` permits widening conversions to occur during a get or set access operation, but throws an `IllegalArgumentException` if a narrowing conversion would occur.

Modifier and Type	Method and Description
`boolean`	`equals(Object obj)` Compares this `Field` against the specified object.
`Object`	`get(Object obj)` Returns the value of the field represented by this `Field`, on the specified object.
`AnnotatedType`	`getAnnotatedType()` Returns an AnnotatedType object that represents the use of a type to specify the declared type of the field represented by this Field.
`<T extends Annotation> T`	`getAnnotation(Class<T> annotationClass)` Returns this element's annotation for the specified type if such an annotation is *present*, else null.
`<T extends Annotation> T[]`	`getAnnotationsByType(Class<T> annotationClass)` Returns annotations that are *associated* with this element.
`boolean`	`getBoolean(Object obj)` Gets the value of a static or instance `boolean` field.
`byte`	`getByte(Object obj)` Gets the value of a static or instance `byte` field.
`char`	`getChar(Object obj)` Gets the value of a static or instance field of type `char` or of another primitive type convertible to type `char` via a widening conversion.
`Annotation[]`	`getDeclaredAnnotations()` Returns annotations that are *directly present* on this element.
`Class<?>`	`getDeclaringClass()` Returns the `Class` object representing the class or interface that declares the field represented by this `Field` object.
`double`	`getDouble(Object obj)` Gets the value of a static or instance field of type `double` or of another primitive type convertible to type `double` via a widening conversion.
`float`	`getFloat(Object obj)` Gets the value of a static or instance field of type `float` or of another primitive type convertible to type `float` via a widening conversion.
`Type`	`getGenericType()` Returns a `Type` object that represents the declared type for the field represented by this `Field` object.
`int`	`getInt(Object obj)` Gets the value of a static or instance field of type `int` or of another primitive type convertible to type `int` via a widening conversion.
`long`	`getLong(Object obj)` Gets the value of a static or instance field of type `long` or of another primitive type convertible to type `long` via a widening conversion.
`int`	`getModifiers()` Returns the Java language modifiers for the field represented by this `Field` object, as an integer.
`String`	`getName()` Returns the name of the field represented by this `Field` object.
`short`	`getShort(Object obj)` Gets the value of a static or instance field of type `short` or of another primitive type convertible to type `short` via a widening conversion.
`Class<?>`	`getType()` Returns a `Class` object that identifies the declared type for the field represented by this `Field` object.
`int`	`hashCode()` Returns a hashcode for this `Field`.
`boolean`	`isEnumConstant()` Returns `true` if this field represents an element of an enumerated type; returns `false` otherwise.
`boolean`	`isSynthetic()` Returns `true` if this field is a synthetic field; returns `false` otherwise.
`void`	`set(Object obj, Object value)` Sets the field represented by this `Field` object on the specified object argument to the specified new value.
`void`	`setBoolean(Object obj, boolean z)` Sets the value of a field as a `boolean` on the specified object.
`void`	`setByte(Object obj, byte b)` Sets the value of a field as a `byte` on the specified object.
`void`	`setChar(Object obj, char c)` Sets the value of a field as a `char` on the specified object.
`void`	`setDouble(Object obj, double d)` Sets the value of a field as a `double` on the specified object.
`void`	`setFloat(Object obj, float f)` Sets the value of a field as a `float` on the specified object.
`void`	`setInt(Object obj, int i)` Sets the value of a field as an `int` on the specified object.
`void`	`setLong(Object obj, long l)` Sets the value of a field as a `long` on the specified object.
`void`	`setShort(Object obj, short s)` Sets the value of a field as a `short` on the specified object.
`String`	`toGenericString()` Returns a string describing this `Field`, including its generic type.
`String`	`toString()` Returns a string describing this `Field`.

- **Methods inherited from class java.lang.reflect.AccessibleObject**

  `getAnnotations`, `getDeclaredAnnotation`, `getDeclaredAnnotationsByType`, `isAccessible`, `isAnnotationPresent`, `setAccessible`, `setAccessible`

- **Methods inherited from class java.lang.Object**

  `clone`, `finalize`, `getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Class Method

- java.lang.Object
  - java.lang.reflect.AccessibleObject
    - java.lang.reflect.Executable
      - java.lang.reflect.Method

- All Implemented Interfaces:
  AnnotatedElement, GenericDeclaration, Member

Modifier and Type	Method and Description
boolean	equals(Object obj) Compares this Method against the specified object.
AnnotatedType	getAnnotatedReturnType() Returns an AnnotatedType object that represents the use of a type to specify the return type of the method/constructor represented by this Executable.
<T extends Annotation> T	getAnnotation(Class<T> annotationClass) Returns this element's annotation for the specified type if such an annotation is *present*, else null.
Annotation[]	getDeclaredAnnotations() Returns annotations that are *directly present* on this element.
Class<?>	getDeclaringClass() Returns the Class object representing the class or interface that declares the executable represented by this object.
Object	getDefaultValue() Returns the default value for the annotation member represented by this Method instance.
Class<?>[]	getExceptionTypes() Returns an array of Class objects that represent the types of exceptions declared to be thrown by the underlying executable represented by this object.
Type[]	getGenericExceptionTypes() Returns an array of Type objects that represent the exceptions declared to be thrown by this executable object.
Type[]	getGenericParameterTypes() Returns an array of Type objects that represent the formal parameter types, in declaration order, of the executable represented by this object.
Type	getGenericReturnType() Returns a Type object that represents the formal return type of the method represented by this Method object.
int	getModifiers() Returns the Java language modifiers for the executable represented by this object.
String	getName() Returns the name of the method represented by this Method object, as a String.
Annotation[][]	getParameterAnnotations() Returns an array of arrays of Annotations that represent the annotations on the formal parameters, in declaration order, of the Executable represented by this object.
int	getParameterCount() Returns the number of formal parameters (whether explicitly declared or implicitly declared or neither) for the executable represented by this object.
Class<?>[]	getParameterTypes() Returns an array of Class objects that represent the formal parameter types, in declaration order, of the executable represented by this object.
Class<?>	getReturnType() Returns a Class object that represents the formal return type of the method represented by this Method object.
TypeVariable<Method>[]	getTypeParameters() Returns an array of TypeVariable objects that represent the type variables declared by the generic declaration represented by this GenericDeclaration object, in declaration order.
int	hashCode() Returns a hashcode for this Method.
Object	invoke(Object obj, Object... args) Invokes the underlying method represented by this Method object, on the specified object with the specified parameters.
boolean	isBridge() Returns true if this method is a bridge method; returns false otherwise.
boolean	isDefault() Returns true if this method is a default method; returns false otherwise.
boolean	isSynthetic() Returns true if this executable is a synthetic construct; returns false otherwise.
boolean	isVarArgs() Returns true if this executable was declared to take a variable number of arguments; returns false otherwise.
String	toGenericString() Returns a string describing this Method, including type parameters.
String	toString() Returns a string describing this Method.

- **Methods inherited from class java.lang.reflect.Executable**

  getAnnotatedExceptionTypes, getAnnotatedParameterTypes, getAnnotatedReceiverType, getAnnotationsByType, getParameters

- **Methods inherited from class java.lang.reflect.AccessibleObject**

  getAnnotations, getDeclaredAnnotation, getDeclaredAnnotationsByType, isAccessible, isAnnotationPresent, setAccessible, setAccessible

- **Methods inherited from class java.lang.Object**

  clone, finalize, getClass, notify, notifyAll, wait, wait, wait

- **Methods inherited from interface java.lang.reflect.AnnotatedElement**

  getAnnotations, getDeclaredAnnotation, getDeclaredAnnotationsByType, isAnnotationPresent

Class Constructor<T>

- java.lang.Object
  - java.lang.reflect.AccessibleObject
    - java.lang.reflect.Executable
      - java.lang.reflect.Constructor<T>

- Type Parameters:
  T - the class in which the constructor is declared

  All Implemented Interfaces:
  AnnotatedElement, GenericDeclaration, Member

Modifier and Type	Method and Description
boolean	equals(Object obj) Compares this Constructor against the specified object.
AnnotatedType	getAnnotatedReceiverType() Returns an AnnotatedType object that represents the use of a type to specify the receiver type of the method/constructor represented by this Executable object.
AnnotatedType	getAnnotatedReturnType() Returns an AnnotatedType object that represents the use of a type to specify the return type of the method/constructor represented by this Executable.
<T extends Annotation> T	getAnnotation(Class<T> annotationClass) Returns this element's annotation for the specified type if such an annotation is *present*, else null.
Annotation[]	getDeclaredAnnotations() Returns annotations that are *directly present* on this element.
Class<T>	getDeclaringClass() Returns the Class object representing the class or interface that declares the executable represented by this object.
Class<?>[]	getExceptionTypes() Returns an array of Class objects that represent the types of exceptions declared to be thrown by the underlying executable represented by this object.
Type[]	getGenericExceptionTypes() Returns an array of Type objects that represent the exceptions declared to be thrown by this executable object.
Type[]	getGenericParameterTypes() Returns an array of Type objects that represent the formal parameter types, in declaration order, of the executable represented by this object.
int	getModifiers() Returns the Java language modifiers for the executable represented by this object.
String	getName() Returns the name of this constructor, as a string.
Annotation[][]	getParameterAnnotations() Returns an array of arrays of Annotations that represent the annotations on the formal parameters, in declaration order, of the Executable represented by this object.
int	getParameterCount() Returns the number of formal parameters (whether explicitly declared or implicitly declared or neither) for the executable represented by this object.
Class<?>[]	getParameterTypes() Returns an array of Class objects that represent the formal parameter types, in declaration order, of the executable represented by this object.
TypeVariable<Constructor<T>>[]	getTypeParameters() Returns an array of TypeVariable objects that represent the type variables declared by the generic declaration represented by this GenericDeclaration object, in declaration order.
int	hashCode() Returns a hashcode for this Constructor.
boolean	isSynthetic() Returns true if this executable is a synthetic construct; returns false otherwise.
boolean	isVarArgs() Returns true if this executable was declared to take a variable number of arguments; returns false otherwise.
T	newInstance(Object... initargs) Uses the constructor represented by this Constructor object to create and initialize a new instance of the constructor's declaring class, with the specified initialization parameters.
String	toGenericString() Returns a string describing this Constructor, including type parameters.
String	toString() Returns a string describing this Constructor.

- **Methods inherited from class java.lang.reflect.Executable**

  getAnnotatedExceptionTypes, getAnnotatedParameterTypes, getAnnotationsByType, getParameters

- **Methods inherited from class java.lang.reflect.AccessibleObject**

  getAnnotations, getDeclaredAnnotation, getDeclaredAnnotationsByType, isAccessible, isAnnotationPresent, setAccessible, setAccessible

- **Methods inherited from class java.lang.Object**

  clone, finalize, getClass, notify, notifyAll, wait, wait, wait

- **Methods inherited from interface java.lang.reflect.AnnotatedElement**

  getAnnotations, getDeclaredAnnotation, getDeclaredAnnotationsByType, isAnnotationPresent