**Runtime type checking**

A variable of a base class can always refer to an object of a subclass. We can determine the actual type of the referred object at runtime.

Java provides several ways to do it:

* the instanceof operator that can be used for testing if an object is of a specified type;
* **java reflection** that can be used to obtain an object representing the class.

The binary operator instanceof returns true if an object is an instance of a particular class or its subclass.

obj instanceof Class

Shape circle = new Circle();  // the reference is Shape, the object is Circle  
Shape rect = new Rectangle(); // the reference is Shape, the object is Rectangle

boolean circleIsCircle = circle instanceof Circle; // true  
boolean circleIsRectangle = circle instanceof Rectangle; // false  
boolean circleIsShape = circle instanceof Shape; // true  
   
boolean rectIsRectangle = rect instanceof Rectangle; // true  
boolean rectIsCircle = rect instanceof Circle; // false  
boolean rectIsShape = rect instanceof Shape; // true

Each object has a method getClass that can be used to obtain an object representing the class. We can directly compare the classes represented by objects at runtime using **java reflection**.

Shape circle = new Circle();

boolean equalsCircle = circle.getClass() == Circle.class; // true  
boolean equalsShape = circle.getClass() == Shape.class;   // false  
boolean rectangle = circle.getClass() == Rectangle.class; // false

There is also another way to check types. An object representing the class has a method isInstance that is similar to the instanceof keyword.

boolean isIntanceOfCircle = Circle.class.isInstance(circle); // true  
boolean isInstanceOfShape = Shape.class.isInstance(circle); // true  
boolean isInstanceOfRectangle = Rectangle.class.isInstance(circle); // false

When to use

Shape shape = new Circle();  
  
if (shape.getClass() == Circle.class) {  
    Circle circle = (Circle) shape;  
   
    // now we can process it as a circle  
}