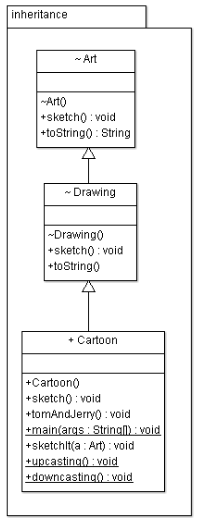
**Cartoon UML**



1. Each of the constructors just outputs “NAME\_OF\_CLASS constructor” e.g. *Art()* outputs “Art constructor”.
2. *sketch()* is a polymorphic instance method i.e. it is coded in the parent class (*Art*) and overridden in each of the subtypes. The methods just output “NAME\_OF\_CLASS::*sketch()*” e.g. in the *Art* class, the *sketch()* method outputs “*Art::sketch()*”.
3. Each of the *toString()* methods return “NAME\_OF\_CLASS::*toString()* e.g. in the *Art* class, the toString() method returns “*Art::toString()*”.
4. *Cartoon* methods:
   1. the *tomAndJerry()* method just outputs “*Cartoon::tomAndJerry()*”.
   2. the *sketchIt()* method takes in an *Art* reference and invokes the *sketch()* method;
      1. try and invoke the *tomAndJerry()* method from within sketchIt(). What happens and why? Fix this so that you can **safely** invoke the *tomAndJerry()* method.
   3. *main()* method
      1. create a *Cartoon* object and refer to it via a *Cartoon* reference namely *c*. Observe the output and especially the **order** of the output.
      2. call the *sketchIt()* method using *c* and dynamically pass down an *Art* object; observe the output closely
      3. call the *sketchIt()* method using *c* and dynamically pass down a *Drawing* object; observe the output closely
      4. call the *sketchIt()* method using *c* and dynamically pass down a *Cartoon* object; observe the output closely
      5. invoke the *upcasting()* method
      6. invoke the *downcasting()* method
   4. upcasting() method
      1. create an *Art* object and refer to it via an *Art* reference namely *a1*. Observe the output. Pass the reference to a *System.out.println()* and observe the output.
      2. create a *Drawing* object and refer to it via an *Art* reference namely *a2*. Observe the output (especially the order). Pass the reference to a *System.out.println()* and observe the output.
      3. create a *Cartoon* object and refer to it via an *Art* reference namely *a3*. Observe the output (especially the order). Pass the reference to a *System.out.println()* and observe the output.
      4. create a *Drawing* object and refer to it via a *Drawing* reference namely *d1*. Observe the output (especially the order). Pass the reference to a *System.out.println()* and observe the output.
      5. create a *Cartoon* object and refer to it via a *Drawing* reference namely *d2*. Observe the output (especially the order). Pass the reference to a *System.out.println()* and observe the output.
      6. create a *Cartoon* object and refer to it via a *Cartoon* reference namely *c1*. Observe the output (especially the order). Pass the reference to a *System.out.println()* and observe the output - why is this one different?
   5. downcasting() method
      1. create a *Drawing* object and refer to it via an *Art* reference namely *a1*. Observe the output (especially the order).
         1. Assign *a1* into a *Drawing* reference *d1*? Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
         2. Pass the reference *d1* to a *System.out.println()* and observe the output.
      2. create a *Cartoon* object and refer to it via an *Art* reference namely *a2*. Observe the output (especially the order).
         1. Assign *a2* into a *Drawing* reference *d2*? Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
         2. Pass the reference *d2* to a *System.out.println()* and observe the output.
         3. Assign *a2* into a *Cartoon* reference *c1*? Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
         4. Pass the reference *c1* to a *System.out.println()* and observe the output.
      3. create a *Cartoon* object and refer to it via a *Drawing* reference namely *d3*. Observe the output (especially the order).
         1. Assign *d3* into a *Cartoon* reference *c2*? Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
         2. Pass the reference *c2* to a *System.out.println()* and observe the output.
      4. create an *Art* object and refer to it via a *Drawing* reference namely *d4*. Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
      5. create an *Art* object and refer to it via a *Cartoon* reference namely *c3*. Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?
      6. create a *Drawing* object and refer to it via a *Cartoon* reference namely *c4*. Do you need a cast? If so, why and will you get a *ClassCastException* at runtime?