<u>EECS1720</u>
<u>Worksheet 4 – More on Polymorphism</u>

1) How does java achieve a form of "multiple inheritance"?

   Does not allow extension from multiple classes. Java allows a similar mechanic through interfaces only. A java class may implement multiple interfaces.

   This is essentially restricting the class to "assume" only method signatures from multiple interfaces, which then need to be implemented. Not quite the same as multiple inheritance via classes which would allow all features to be assumed by the extending class (including already defined fields and methods from the parent)

2) What is a declared type?

   The type we specify for a reference variable that we create (declaration prior to assignment)

```
i.e.   Type myVar ;
or     Type myVar = someValue;    // LHS defines type declared for "myVar"

e.g.   String myVar;              // declared type of myVar is String
       myVar = "hello";
       Object myVar2 = myVar;     // declared type of myVar2 is Object
```

3) What is a run-time/actual type?

   The type of the object we <u>assign</u> to a given reference at any given instant during the execution of a program (run-time).

   In the above example for (12), myVar2 is a reference to an Object type, however it is assigned a String type object at runtime, since Object is a root class for all other objects, String "is-a" Object, so a string object at runtime may be assigned to myVar2 (substitutability principle).

   At runtime, myVar2 has a "run-time/actual" type of String (in the above example).

4) What is early binding?

   At compile time (which occurs early ... before program is run), the type (and thus accessible fields/methods) that are available through an object reference are bound to the reference and these are what may be accessed/invoked through that reference

5) What is late binding?

At run-time (late .. while program is running), the methods that <u>are actually invoked</u> through an object reference are bound to the method call (invocation) – this depends both on the run-time type, and the signature (arguments) a method is invoked with.

Overloaded & Overridden methods are bound to the method call at run-time due to the run-time type assigned to a given reference, and the arguments of that method call.

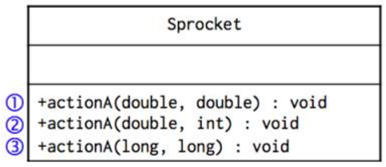
6) What is dynamic dispatch?

dynamic dispatch is the process of late binding - how the JVM knows which version of a method to run or which attribute to bind to an object reference at run-time depending on what run-time type is currently assigned to the reference.

7) Given a method invocation, the JVM performs late **<u>binding</u>** with the method with the most specific signature. For each of the following method invocations indicate which method has the most specific signature

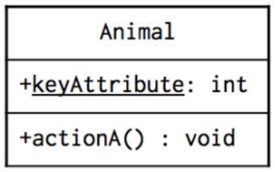
Declared type and run-time type are both Sprocket

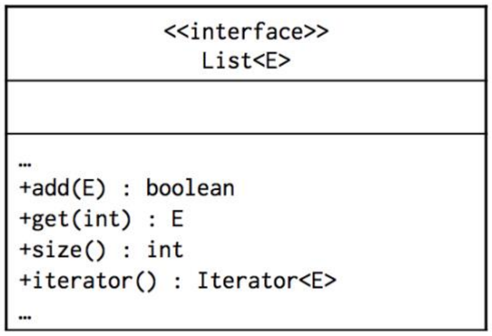
```
Sprocket s = new Sprocket();
```

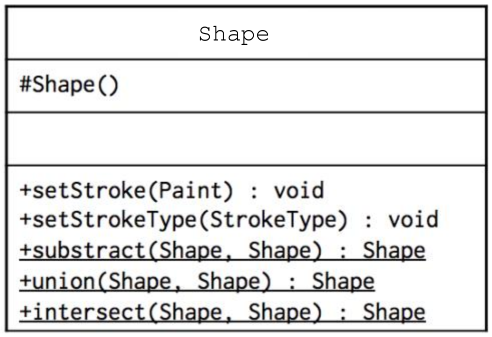

Sprocket
① +actionA(double, double) : void
② +actionA(double, int) : void
③ +actionA(long, long) : void

```
// Q1
s.actionA(99.9,33.3);   // e.g. this method invocation binds with??   actionA(double, double)
// Q2
s.actionA(99,33.3);         actionA(double, double)
// Q3
s.actionA(99L,33L);         actionA(long, long)
// Q4
s.actionA(99L,33.3);        actionA(double, double)
// Q5
s.actionA(99f,33f);         actionA(double, double)
// Q6
s.actionA(99f,(short) 33);      actionA(double, int)
```

8) Notation? Identify the fields, methods, whether they are static, abstract, what access they have, what their signature is (if a method), and whether one can instantiate or not.






Shape
#Shape()
+setStroke(Paint) : void
+setStrokeType(StrokeType) : void
+substract(Shape, Shape) : Shape
+union(Shape, Shape) : Shape
+intersect(Shape, Shape) : Shape

Static fields: Animal.keyAttribute -> public

Ctors: Animal -> no ctor defined, so Animal() assumed (can instantiate)
Shape() -> protected (can only be called by subclasses, so cannot be instantiated as a Shape object, but concrete child classes can)

Methods:

Animal:

actionA() -> non static and public

Shape: (all methods public)

setStroke(Paint), setStrokeType(StrokeType) -> non static
substract(Shape, Shape):Shape -> static
union(Shape, Shape):Shape -> static
intersect(Shape,Shape):Shape -> static

List<E> interface: (all methods public, signatures as in diagram)
Cannot instantiate as List<E> is an interface

9) What is meant by polymorphism?

When an object (reference) exhibits different form / behaviour at different times during execution.

10) Why would you declare a reference variable to be higher up the hierarchy than an object you instantiate?

To enable polymorphic behaviour.

E.g. to invoke a method that is available in the parent, but overridden in several child classes. This way a common reference type (or set of references in a collection) could be declared and methods invoked, but each reference could refer to different “actual” objects (specialized versions) that each have different behaviour despite the same method being invoked