CS558 Programming Languages – Fall 2023 – Study Questions Lecture 10a

1. Consider the code on slide 11.

(a) Suppose we change the first line of `Main` to

```
val animals : List[Animal] = List(new Bird(), new Lion())
```

and add one of the following class definitions. For each definition, indicate if the resulting code is valid in Scala. If the code is not valid, indicate what kind of error occurs (static or runtime) and where.

(i)
```
class Lion extends Animal {
    def name() = "lion"
}
```

(ii)
```
class Lion extends Animal {
    def name() = "lion"
    def eat() = "chomps red meat"
    def speak() = "roar"
}
```

(iii)
```
class Lion {
    def name() = "lion"
    def eat() = "chomps red meat"
}
```

(b) Now consider the following variation on the slide 11 code, in similar syntax, but intended to be in a *dynamically typed* language rather than in Scala.

```
class Animal {
  def name() = "default name"
  def eat() = "default eat"
}

class Bird extends Animal {
  def name() = "bird"
  def eat() = "chomp on insects"
}

object Main {
  val animals = List(new Bird(),new Lion())
  for (animal <- animals)
     println(animal.name() + ":" + animal.eat())
}
```

For each of the following possible definitions of class `Lion`, indicate if the resulting code would run without errors. For those that would not, indicate what kind of error would occur, and where.

(i)
```
class Lion extends Animal {
    def name() = "lion"
}
```

(ii)
```
class Lion extends Animal {
    def name() = "lion"
    def eat() = "chomps red meat"
    def speak() = "roar"
}
```

(iii)
```
class Lion {
    def name() = "lion"
    def eat() = "chomps red meat"
}
```

(iv)
```
class Lion {
    def name() = "lion"
    def speak() = "roar"
}
```

2. Consider this Java program:

```
class Q1 {
  int f() {return 1;}
  int g() {return f();}
}

class Q2 extends Q1 {
  int f() {return 2;}
}

class B {
  public static void main(String argv[]) {
    Q1 a = new Q1();
    System.out.println(a.g());
    Q2 b = new Q2();
    System.out.println(b.g());
    Q1 c = new Q2();
    System.out.println(c.g());
    Q2 d = (Q2) (new Q1());  // cast operation
    System.out.println(d.g());
  }
}
```

What happens when we execute `java B`? Why?

3.(a) Consider the code on slide 22. Explain why the fact that `x` and `y` have the same slot positions in objects of classes `A` and `B` is essential for making the call `b.f()` work properly.

(b) Now consider this slight variant of the code:

```
class A {
  int x;
  int y;
  f() = x+y;
  g() = f()
}

class B extends A {
  int z;
  f() = x+z;
}
val a:A = ...
val b:B = ...
val w = a.g() + b.g()
```

Write down low-level code in the style of slide 22 for `A.f`, `B.f`, `g` and `w`. Show how the layout of `A` and `B` objects and classes (vtables) should change for this code, in order to give semantics consistent with those described on slide 14.

(c) Now consider the following code:

```
class A  {
  int x;
  f() = x;
}

class B {
  int y;
  g() = y;
}

class C extends A,B { // multiple inheritance
}

val c:C = ...
val v = c.f() + c.g()
```

Explain why the offset-sharing representation used on slides 21-22 will not work for representing `C`, due to the multiple inheritance. (Attempting to give low-level code for `f`, `g`, and `v` may help expose the issue.)