Montgomery College, CMSC 203
Worksheet 1
Module 17

Objectives
- Interfaces
- Polymorphism with interfaces

Concept Questions

1) In Java, a(n) ____________________ is a collection of constants and abstract methods.

   a) polymorphic reference
   b) abstract class
   c) implementation
   d) interface
   e) iterator

   Answer: d

2) Write a header for an interface called "Animal"

   Answer: `public interface Animal`

3) The fields in the interfaces are treated as:

   a) final
   b) static
   c) both a and b
   d) interfaces cannot contain fields

   Answer: c

4) (True/False) An instance of an interface CAN be created just like an instance of a class.

   Answer: False

5) A class can be derived from (one/multiple) superclass(es) and it can implement (one/multiple) interface(s).

   Answer: one, multiple

6) A polymorphic reference is one that can refer to ________________ type(s) of object(s).

   a) exactly one
   b) zero
   c) multiple
   d) abstract
   e) static

   Answer: c

7) In Java, polymorphic references can be created through the use of ___________________ and _________________.

a) inheritance, interfaces
b) inheritance, abstract classes
c) interfaces, abstract classes
d) interfaces, iterators
e) none of the above

Answer: a

8) Suppose `Animal` is an interface that specifies a single method – `speak`. Now suppose the Dog class implements the `Animal` interface. In addition to the speak method, the Dog class also has a method called `wagTail`. Now consider the following code.

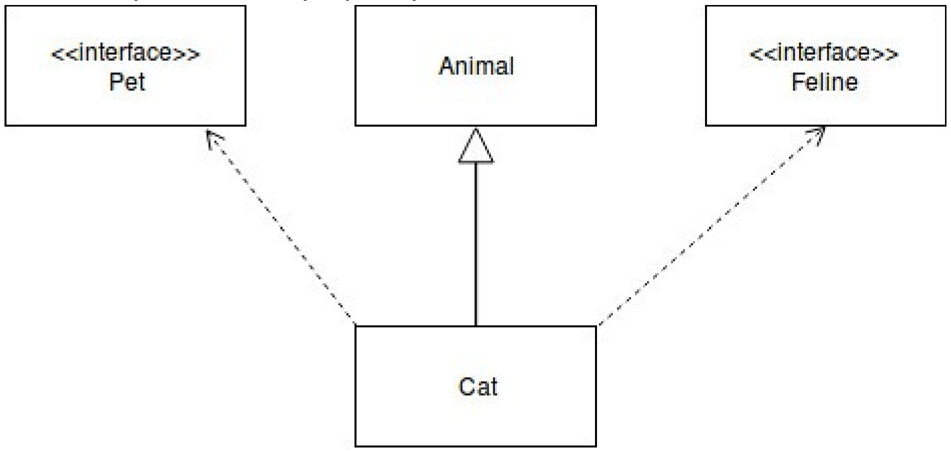
```
Animal a = new Dog();
a.wagTail();
```

Which of the following is true about this code?

a) It will result in a compile-time error.
b) It will result in a run-time error.
c) It will call the speak method defined in the Animal interface.
d) It will call the `wag`Tail method defined in the Dog class.
e) none of the above are true.

Answer: a

9) Write a header that represents the polymorphic architecture of the class `Cat`:



Answer: `public class Cat extends Animal implements Pet, Feline`

10) It is possible to define a method in the interface by using a:

a) static method
b) final static method
c) default method

d) you cannot define methods in the interface

Answer: c

11) Methods in an interface have public visibility by default. (**True**/False)

12) All the methods in the Interface are abstract by default (**True**/False)

13) What is the wrong with the following code?(assume each class is defined in its own java file).

```
1. public interface MobileDevice
2. {
3.   String MNUFACTURE;
4.    public String turnOn();
5.    public String takePicture() { return "Ready to take picture"; }
6.    public String record(int start, int end);
7.    public String pause();
8. }
```

```
1. public class Iphone implements MobileDevice {
2.   public String turnOn () { return "Iphone is turned on"; }
3.   public String takePicture () { return "picture taken by iphone"; }
4.   public String pause(){ return "pause recording"; }
5. }
```

Answer:

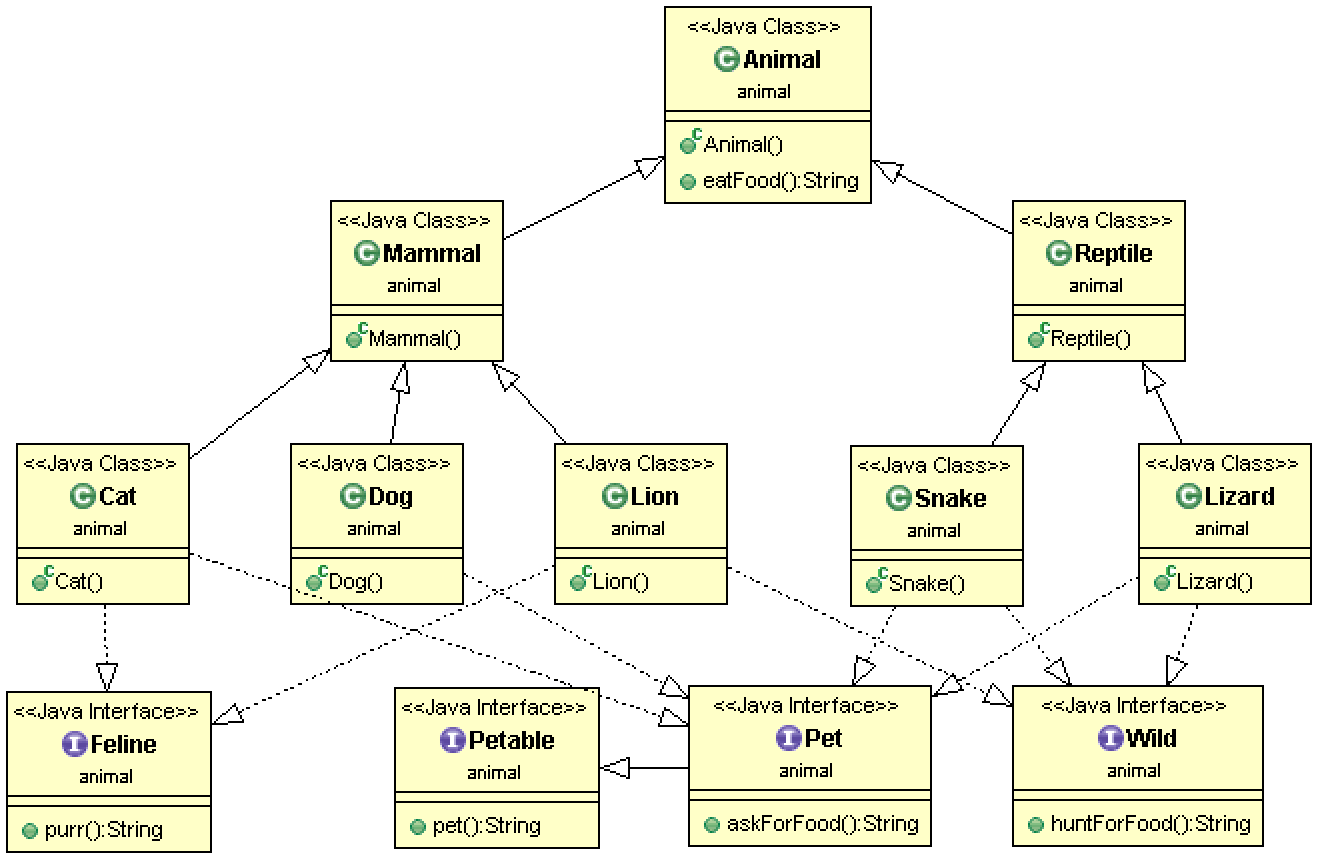
Line 3 : MANUFACTURE should be initialized because it is final Staitc by default and all the static fields need to be initialized.

Line 5 : method of the interface cannot have body unless it is defined as `default`.

Line 1: will cause compilation error because the method `record` is not implemented in the `Iphone` Class.

Programming Question:

1. Convert the following UML diagram design into classes and interfaces



2. Implement the methods seen in the UML diagram. All the methods simply return a string of the activity. For example, askForFood() method implementation in the Cat subclass will simply return a string "Cat is asking for food". Another example is a pet() method in the Petable interface will return "Being petted".

Answer:

```
public class Animal {
	public String eatFood(){
		return "Eating food";
	}
}

public class Mammal extends Animal {
}

public class Reptile extends Animal {
}

public interface Feline {
	public String purr();
}

public interface Petable {
	public String pet();
}
```

```java
public interface Pet extends Petable {
	public String askForFood();
}

public interface Wild {
	public String huntForFood();
}


public class Cat extends Mammal implements Feline, Pet{

	@Override
	public String pet() {
		return "Cat is being pet";
	}

	@Override
	public String askForFood() {
		return "Cat is asking for food";
	}

	@Override
	public String purr() {
		return "Cat is purring";
	}

}

public class Dog extends Mammal implements Pet {

	@Override
	public String pet() {
		return "Petting dog";
	}

	@Override
	public String askForFood() {
		return "Dog is asking for food";
	}

}

public class Lion extends Mammal implements Feline, Wild {

	@Override
	public String huntForFood() {
		return "Lion is hunting for food";
	}

	@Override
	public String purr() {
		return "Lion is purring";
	}


}
```

```java
public class Snake extends Reptile implements Wild, Pet {

	@Override
	public String pet() {
		return "Petting snake";
	}

	@Override
	public String askForFood() {
		return "Snake is asking for food";
	}

	@Override
	public String huntForFood() {
		return "Snake is hunting for food";
	}

}


public class Lizard extends Reptile implements Pet, Wild {

	@Override
	public String pet() {
		return "Petting lizard";
	}

	@Override
	public String huntForFood() {
		return "Lizard is hunting for food";
	}

	@Override
	public String askForFood() {
		return "Lizard is asking for food";
	}

}
```