CSM 61B Fall 2017

Inheritance

Mentoring 4: September 18, 2017

1 Dogs Yay

1.1
```
class Dog {
    public void walk() {
        System.out.println("The dog is walking");
    }
}
class Beagle extends Dog {
    @Override
    public void walk() {
        System.out.println("The beagle is walking");
    }
}
```

What would Java display?

(a)
```
Dog fido1 = new Dog();
fido1.walk();
```

(b)
```
Beagle fido2 = new Beagle();
fido2.walk();
```

(c)
```
Beagle fido3 = new Dog();
fido3.walk();
```

(d)
```
Dog fido4 = new Beagle();
fido4.walk();
```

1.2 What would each call in `Poodle.main` print? If a line would cause an error, determine if it is a compile-error or runtime-error.

```
class Dog {
    void bark(Dog dog) {
        System.out.println("bark");
    }
}

class Poodle extends Dog {
    void bark(Dog dog) {
        System.out.println("woof");
    }
    void bark(Poodle poodle) {
        System.out.println("yap");
    }
    void play(Dog dog) {
        System.out.println("no");
    }
    void play(Poodle poodle) {
        System.out.println("bowwow");
    }
    public static void main(String[] args) {
        Dog dog = new Poodle();
        Poodle poodle = new Poodle();

        dog.play(dog)
        dog.play(poodle)
        poodle.play(dog)
        poodle.play(poodle)

        poodle.bark(dog)
        poodle.bark(poodle)
        dog.bark(dog)
        dog.bark(poodle)

    }
}
```

2 An Appealing Appetizer

2.1
```
public interface Consumable {
    public void consume();
}
public abstract class Food implements Consumable {
    String name;
    public abstract void prepare();
    public void play() {
        System.out.println("Mom says, 'Don't play with your food.'");
    }
}
public class Snack extends Food {
    public void prepare() {
        System.out.println("Taking " + name + " out of wrapper");
    }
    public void consume() {
        System.out.println("Snacking on " + name);
    }
}
```

(a) Compare and contrast interfaces and abstract classes.

(b) Do we need the `play` method in `Snack`?

(c) Does this compile? `Consumable chips = new Snack();`

3 Iterator Interface

In Java, an **iterator** is an object which allows us to traverse a data structure in linear fashion. Every iterator has two methods: `hasNext` and `next`.

```
interface IntIterator {
    boolean hasNext();
    int next();
}
```

3.1 Consider the following code that demonstrates the `IntArrayIterator`.

```
int[] arr = {1, 2, 3, 4, 5, 6};
IntIterator iter = new IntArrayIterator(arr);
if (iter.hasNext()) {
    System.out.println(iter.next());     // 1
}
if (iter.hasNext()) {
    System.out.println(iter.next() + 3); // 5
}
while (iter.hasNext()) {
    System.out.println(iter.next());     // 3 4 5 6
}
```

Define an `IntArrayIterator` class that works as described above.

3.2 Define an `IntListIterator` class that adheres to the `IntIterator` interface.

3.3 Define a method, `printAll`, that prints every element in an `IntIterator` **regardless of how the iterator is implemented.**

4 Pokemon *Extra Practice*

4.1 Identify the errors that occur when running the code to the right.

```
Pikachu p = new Pikachu();
Pokemon a = p;
p = a;
a.eat();
a = new Squirtle();
a.attack();
((Squirtle) a).attack();
Pokemon z = new Pikachu();
Squirtle s = (Squirtle) z;
((Pokemon) p).attack(z);
```

```
public class Pokemon {
    public int hp, power;
    public String cry;
    public String secret;
    public Pokemon() {
        hp = 50;
        cry = "Poke?";
    }
    public Pokemon(String c, int hp) {
        cry = c;
        this.hp = hp;
    }
    public void attack(Pokemon p) {
        p.hp -= power;
    }
    public void eat() {
        System.out.println("nom nom");
    }
}
public class Pikachu extends Pokemon {
    public Pikachu() {
        hp = 100;
    }
    public Pikachu(int hp) {
        super("Pika pika pikachu", hp);
    }
    public void attack(Pokemon p) {
        p.hp = 0;
    }
    public void eat() {
        System.out.println("nom Pika nom");
    }
}
public class Squirtle extends Pokemon {
    public void attack() {
        System.out.println("Water gun!!");
    }
}
```