

1 Overloading vs. Overriding

Dynamic method selection does not happen for **overloaded** methods.

- When some other class has two methods, one for the supertype and one for the subtype.

```
public class Dog {  
    public void bark(Dog x) { ... }  
    public void bark(Cat x) { ... }  
}
```

Dynamic method selection only happens for **overridden** methods.

- When instance method of subtype overrides some method in supertype.

```
public interface Animal {  
    public void makeNoise();  
}  
public class Pig implements Animal {  
    public void makeNoise() {  
        System.out.print("oink");  
    }  
}
```

2 Casting

Casting is a powerful but dangerous tool.

- Tells Java to treat an expression as having a different compile-time type.
- Effectively tells the compiler to ignore its type checking duties.

```
Poodle frank = new Poodle("Frank", 5);  
Malamute frankSr = new Malamute("Frank Sr.", 100);  
// Poodle largerPoodle = (Poodle) maxDog(frank, frankSr);
```

3 extends keyword

- Variables, methods, nested classes.
- Not constructors.
- Subclass constructor must invoke superclass constructor first.
- Use super to invoke overridden superclass methods and constructors.

4 Pokemon (Daniel Nguyen)

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

```

public class Pokemon {
    protected int hp;
    public String cry;
    private String secret;
    private int power;

    public Pokemon() {
        hp = 50;
        cry = "Poke?";
    }
    public Pokemon(String c) {
        this(c, 50);
    }
    public Pokemon(String c, int hp) {
        cry = c;
        this.hp = hp;
    }
    public void attack(Pokemon p) {
        p.hp -= power;
    }
    public void talk() {
        System.out.println(cry);
    }
    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!!");
    }
}

```

```

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);

```