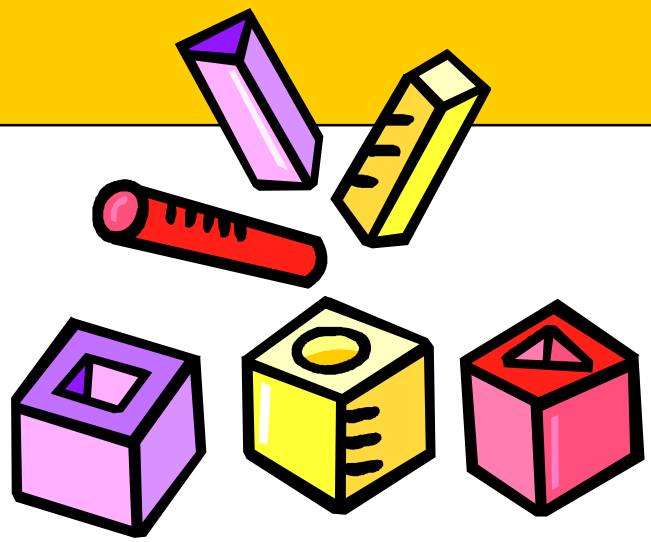
Victoria University
of Wellington, New Zealand
Te Whare Wananga o te
Upoko o te Ika a Maui
Aotearoa

SWEN221:
Software Development

9: Polymorphism I

David J. Pearce & Nicholas Cameron & James Noble & Petra Malik
Computer Science, Victoria University

Polymorphism

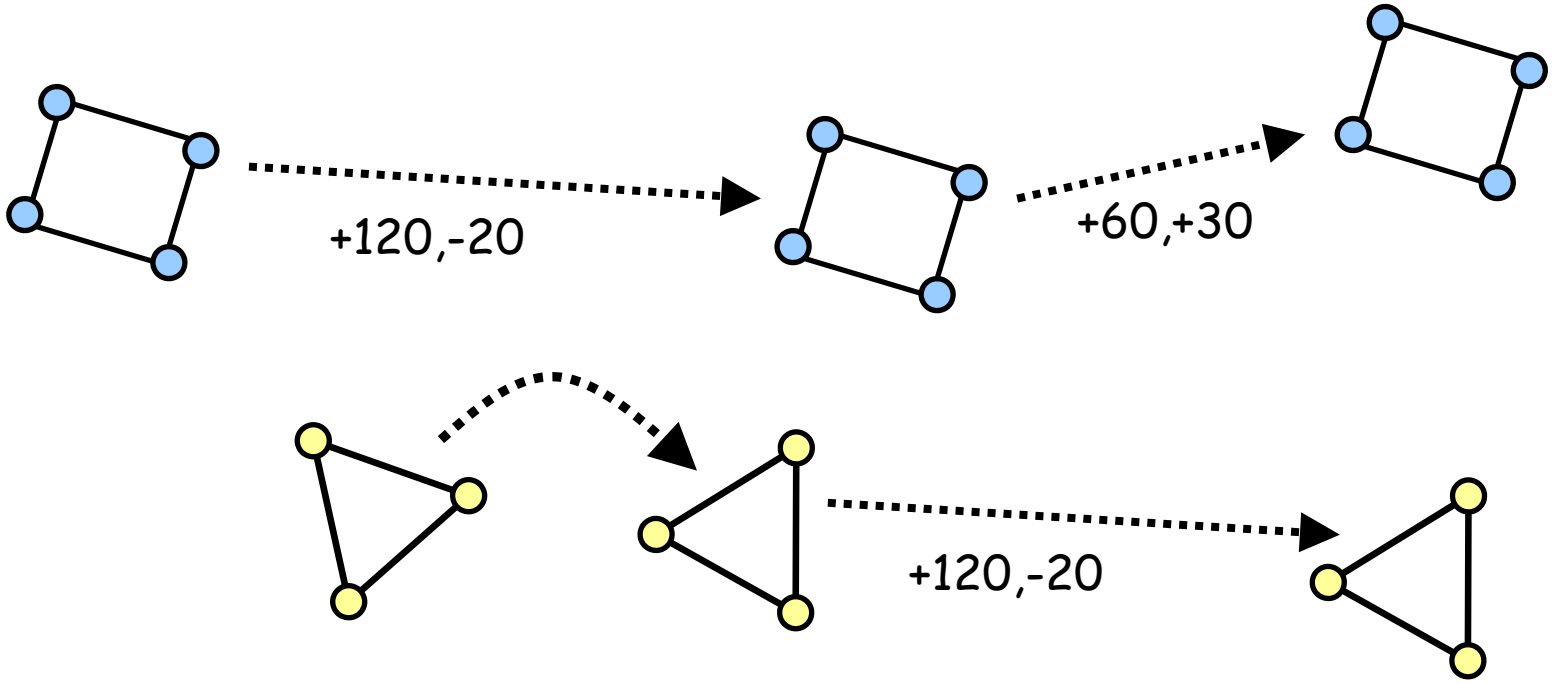


Gk. πολύμορφή

Fr. poly (many),
morphe (shape)

- **Numeric Coercions**
- **Subclass Polymorphism (a.k.a dynamic dispatch)**
- **Overloading**
- Generics — Parametric Polymorphism
  - Generic Classes & Functions

Understanding Polymorphism



- It's all about treating different things in the same way!
  - E.g. method for moving or rotating shapes shouldn't worry about what shape it is!

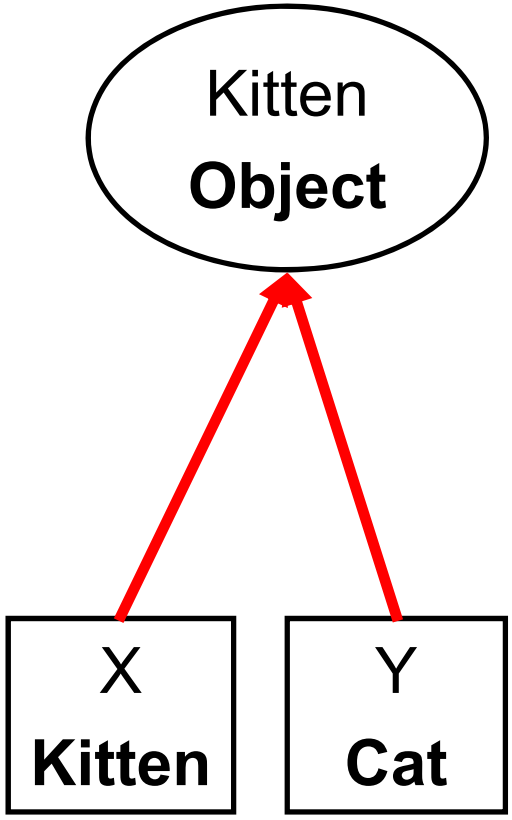
Mental Model of Typing

```
class Cat { … }
class Kitten extends Cat { … }

Kitten x = new Kitten();
Cat y = x;
```

Static (or Declared) type of y is "Cat"

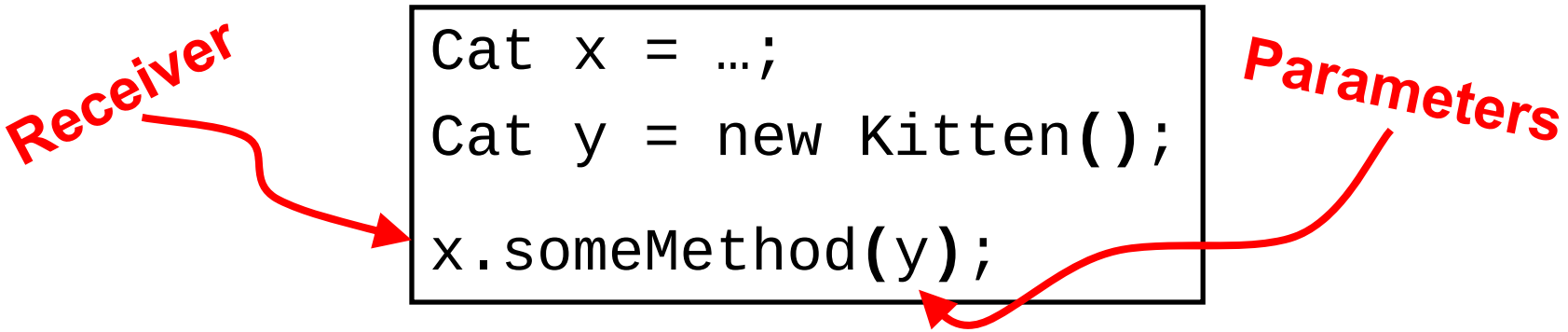
Dynamic (or Runtime) type of x and y is "Kitten"



- Static Type:
  - **Declared** type of a variable
- Dynamic Type:
  - Type of object **referred** to by variable

Dynamic Dispatch

- Dynamic dispatch:
  - The **mechanism** which enables **subclass polymorphism**



- Method dispatch:
  - Two phases – compile time (static) and runtime (dynamic)
- Static checking phase:
  - Based on static types of receiver and parameters
  - Can only call methods defined in static type of receiver
- Dynamic dispatch:
  - Selection of method at runtime
  - Dynamic choice depends only on receiver

Why dynamic dispatch?

```
class HouseCat {
 …                  // lots of methods!
 void speak() { System.out.println(“Meow!”);
}}
class Tiger extends HouseCat {
 void speak() {
  System.out.println(“ROOOOAAARRRR”);
}}
```

- Dynamic dispatch + subclassing
  - Allows one class to be **constructed from another**
  - Useful when **behaviour** of subclass mostly the same
    - E.g. a Tiger behaves like a HouseCat, except it’s LOUDER!
  - Dynamic dispatch ensures subclass **can change behaviour** as needed

Quiz: what gets printed?

```
class Cat {
 String whatAmI() {
  return “I’m a Cat!”;
}}

class Kitten extends Cat {
 String whatAmI() {
  return “I’m a Kitten!”;
}}

Cat gypsy = new Cat();
Cat spike = new Kitten();

System.out.println(“Gypsy: ” + gypsy.whatAmI());
System.out.println(“Spike: ” + spike.whatAmI());
```

A)
```
Gypsy: “I’m a Kitten!”
Spike: “I’m a Kitten!”
```

B)
```
Gypsy: “I’m a Cat!”
Spike: “I’m a Kitten!”
```

C)
```
Gypsy: “I’m a Cat!”
Spike: “I’m a Cat!”
```

More Dispatch Examples

```
class Cat {
 String whatAmI() {
  return “I’m a Cat!”;
}}

class Kitten extends Cat {
 String whatAmI() {
  return “I’m a Kitten!”;
}}

class NinjaKitten extends Kitten {
 String isKickedBy(Kitten k) { return “Ouch!”; }
}

Cat bob = new NinjaKitten();
System.out.println(“Bob: ” + bob.whatAmI());
```

A) Bob: “I’m a Kitten!”

B) Bob: “Ouch!”

C) **error**

More Dispatch Examples

```
class Cat {
 String whatAmI() {
  return “I’m a Cat!”;
}}

class Kitten extends Cat {
 String whatAmI() {
  return “I’m a Kitten!”;
}}

class NinjaKitten extends Kitten {
 String isKicked () { return “Ouch!”; }
}

Cat bob = new NinjaKitten();
System.out.println(“Bob: ” + bob.isKicked());
```

A) Bob: “I’m a Kitten!”

B) Bob: “Ouch!”

C) **error**

More Dispatch Examples

```
class Cat {
 String whatAmI() {
  return “I’m a Cat!”;
 }
 void print() {
  System.out.println(whatAmI());
}}
class Kitten extends Cat {
 String whatAmI() {
  return “I’m a Kitten!”;
}}
Cat gypsy = new Cat();
Cat spike = new Kitten();
gypsy.print();
spike.print();
```

A) “I’m a Kitten!”
“I’m a kitten!”

B) “I’m a Cat!”
“I’m a Kitten!”

C) “I’m a Cat!”
“I’m a Cat!”

Quiz

```
class Cat {
 …
 public void isClawedBy(Cat c) {
  System.out.println(“Clawed by a Cat!”);
}}
class Kitten extends Cat {
 public void isClawedBy(Kitten k) {
  System.out.println(“Clawed by a Kitten!”);
}}
Cat gypsy = new Cat();
Cat spike = new Kitten();
Kitten teddy = new Kitten();
gypsy.isClawedBy(teddy);
spike.isClawedBy(teddy);
teddy.isClawedBy(teddy);
```

A)
```
“Clawed by a Cat!”
“Clawed by a Kitten!”
“Clawed by a Kitten!”
```

B)
```
“Clawed by a Cat!”
“Clawed by a Cat!”
“Clawed by a Kitten!”
```

Summary

- Subclass Polymorphism
  - A product of inheritance
  - Aids reuse
  - Key part of OO
  - Dynamic dispatch
- Next time
  - More on dynamic dispatch…