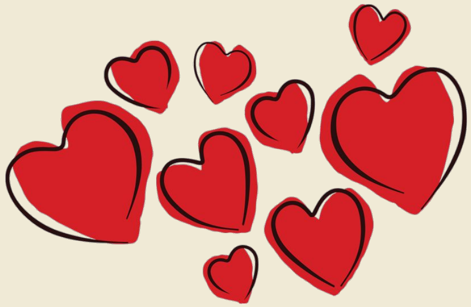
disc. 5
cs61b sp22



test 1 review

method overloading/overriding, casting, DMS

slides
bit.ly/abhi-disc

attendance
bit.ly/abhi-attendance

announcements

announcements

1. HW3 due today

announcements

1. HW3 due today
2. Test 1 on Wednesday, 2/16

announcements

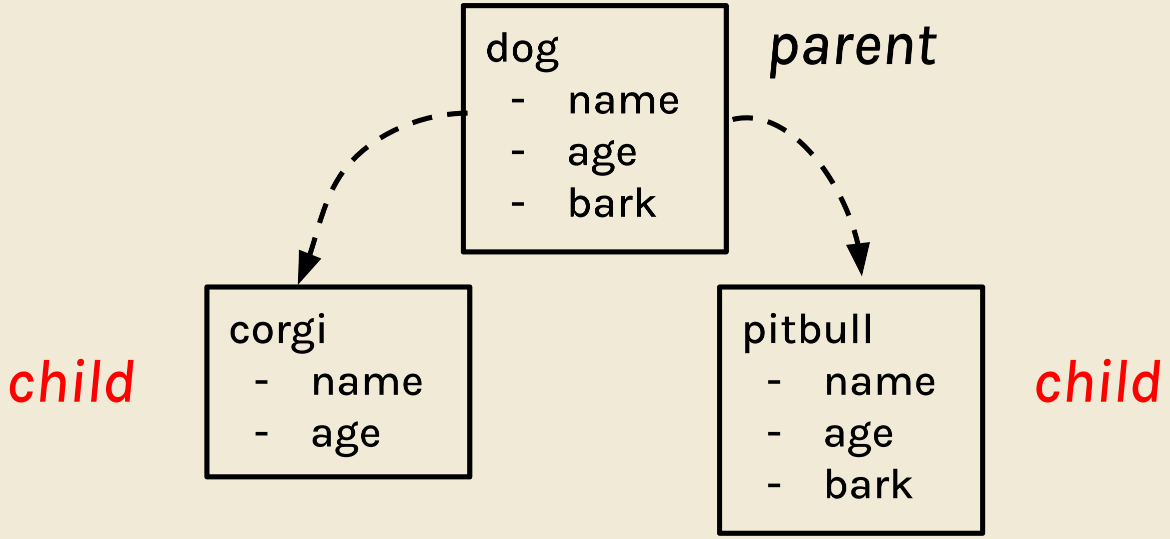
1. HW3 due today
2. Test 1 on Wednesday, 2/16
3. Lab 5 due NEXT Tuesday, 2/22

announcements

1. HW3 due today
2. Test 1 on Wednesday, 2/16
3. Lab 5 due NEXT Tuesday, 2/22
4. Project 1: Enigma RELEASED!
    a. Checkpoint due Friday, 2/25

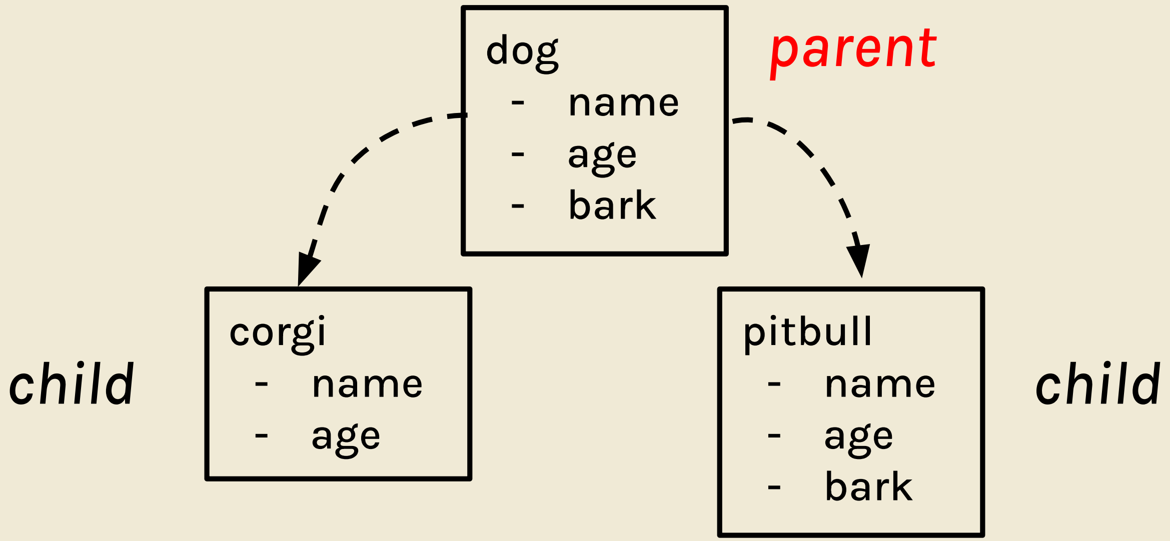
subclasses/child classes (review)

- classes that extend another class



superclasses/parent classes

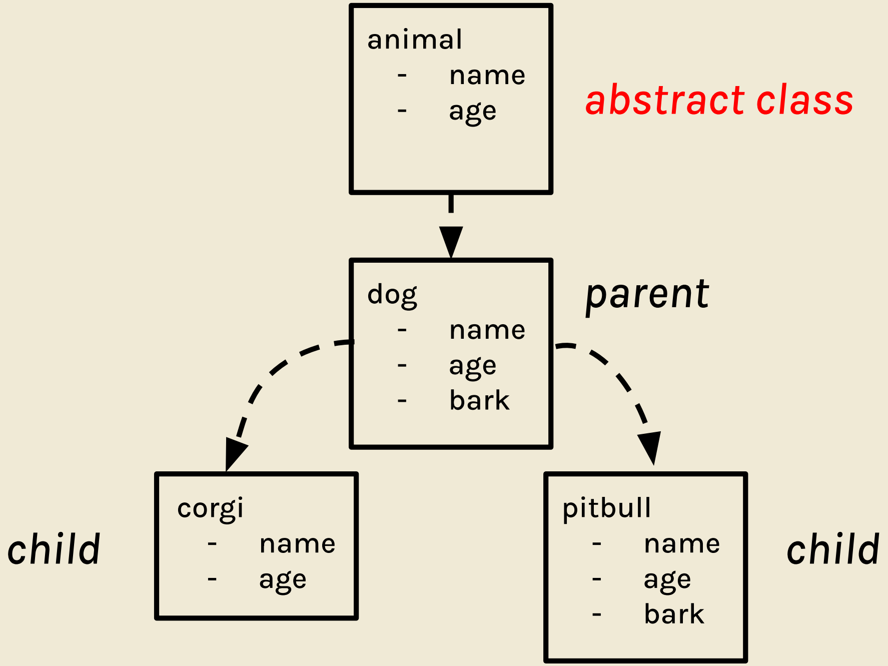
- classes that are extended by other classes



abstract classes (review)

- cannot be directly referenced
  - must be extended by a **concrete class**
  - describe the functions that classes of this “type” should be able to do

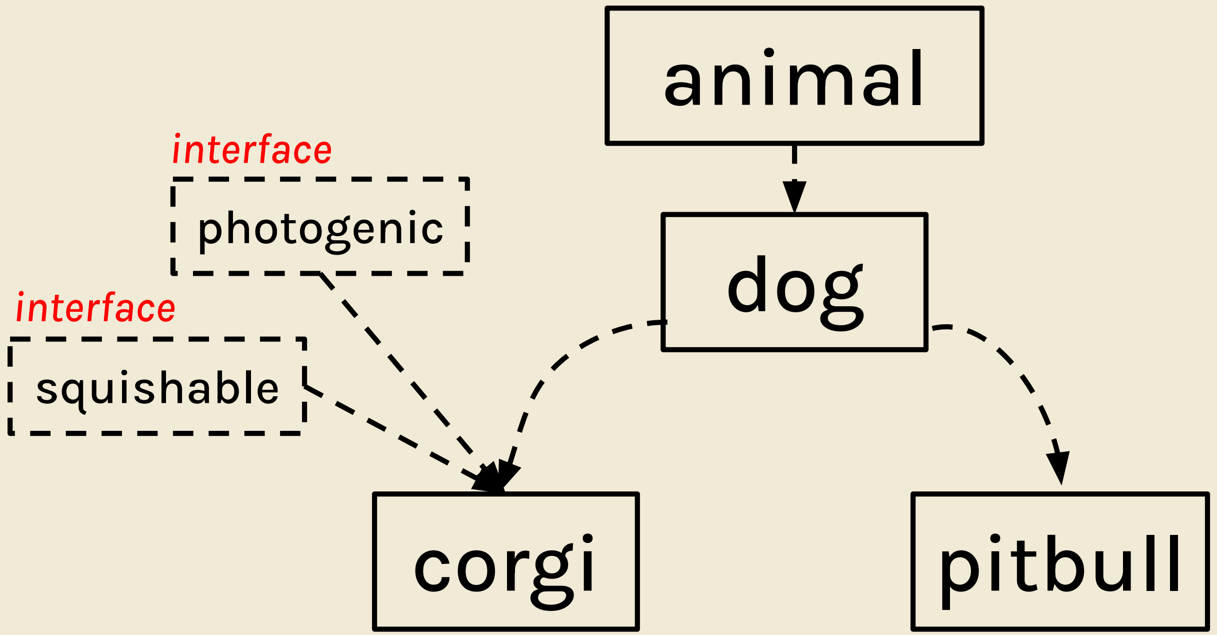
abstract classes (review)



interfaces (review)

- **implemented** by classes
- specify methods that describe an ability
  - e.g., Comparable, List
  - these methods aren’t usually “filled out”—they’re just blueprints for the “implementing” class

interfaces (review)



method overloading

- multiple methods with the same name and return type, but different parameters

method overloading

```
public void barkAt(Dog d) {
    System.out.print(“Woof, it’s another dog!”);
}

public void barkAt(Animal a) {
    System.out.print(“Woof, what is this?”);
}
```

method overriding

- subclass method has same exact signature as superclass method

method overriding

```
In Dog class:
    public void speak() {
        System.out.print(“Woof, I’m a dog!”);
    }

In Corgi Class:
    public void speak() {
        System.out.print(“Woof, I’m a corgi!”);
    }
```

casting

- allows us to call a subclass's method on a variable that's statically typed to be the superclass

```
Animal a = new Dog();

Dog d = a; error
Dog d = (Dog) a; great
```

DMS

dynamic method selection

compile time checks

run time checks

compile time checks

- variable assignments

run time checks

compile time checks

- variable assignments
- method calls (consider only static type)

run time checks

compile time checks

- variable assignments
- method calls (consider only static type)

run time checks

- overridden methods

compile time checks

- variable assignments
- method calls (consider only static type)

run time checks

- overridden methods
- ensure casted objects can be assigned to their variables (consider only dynamic type)

compile time checks

- variable assignments
- method calls (consider only static type)

run time checks

- overridden methods
- ensure casted objects can be assigned to their variables (consider only dynamic type)

fields are **always** chosen based on static type

worksheet

(on 61B website)

1A Classy Cats

```java
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = “Huh?”;
    }
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
}
```

```java
public class Cat extends Animal {




}
```

Fill in the Cat constructor such that it makes a “Meow!” noise when greet() is called (in all caps for kittens).

1A Classy Cats

```java
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + ": " + makeNoise();
    }
}
```

```java
public class Cat extends Animal {
    public Cat(String name, int age) {


    }
}
```

Fill in the Cat constructor such that it makes a "Meow!" noise when greet() is called (in all caps for kittens).

1A Classy Cats

```
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = “Huh?”;
    }
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
}
```

```
public class Cat extends Animal {
    public Cat(String name, int age) {
        super(name, age);

    }
}
```

Fill in the Cat constructor such that it makes a “Meow!” noise when greet() is called (in all caps for kittens).

1A Classy Cats

```java
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = “Huh?”;
    }
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
}
```

```java
public class Cat extends Animal {
    public Cat(String name, int age) {
        super(name, age);
        this.noise = “Meow!”;
    }
}
```

Fill in the Cat constructor such that it makes a “Meow!” noise when greet() is called (in all caps for kittens).

1B Classy Cats

```
public abstract class Animal {
    protected String name;
    protected String noise = “Huh?”;
    protected int age;
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
    public abstract void shout();
    abstract void count(int x);
}
```

```
public class Cat extends Animal {
    public Cat() {...}
    public Cat(String name, int age) {
        this();
        this.name = name;
        this.age = age;
    }
    @Override
    ____________________ {
        System.out.println(...);
    }
    @Override
    ____________________ {
        for(int i = 0; i < x; i++) {
            System.out.println(...);
        }
    }
}
```

Make the Cat class compatible with Animal

1B Classy Cats

```
public abstract class Animal {
    protected String name;
    protected String noise = “Huh?”;
    protected int age;
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
    public abstract void shout();
    abstract void count(int x);
}
```

```
public class Cat extends Animal {
    public Cat() {...}
    public Cat(String name, int age) {
        this();
        this.name = name;
        this.age = age;
    }
    @Override
    public void shout() {
        System.out.println(...);
    }
    @Override
    ____________________ {
        for(int i = 0; i < x; i++) {
            System.out.println(...);
        }
    }
}
```

Make the Cat class compatible with Animal

1B Classy Cats

```
public abstract class Animal {
    protected String name;
    protected String noise = “Huh?”;
    protected int age;
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + “: “ + makeNoise();
    }
    public abstract void shout();
    abstract void count(int x);
}
```

```
public class Cat extends Animal {
    public Cat() {...}
    public Cat(String name, int age) {
        this();
        this.name = name;
        this.age = age;
    }
    @Override
    public void shout() {
        System.out.println(...);
    }
    @Override
    void count(int x) {
        for(int i = 0; i < x; i++) {
            System.out.println(...);
        }
    }
}
```

Make the Cat class compatible with Animal

2A The Interfacing CatBus

```
__________________________________________ {

     ________________________ {
          System.out.println(“Purrrrrrr”);
     }

     ___________________ {
          System.out.println(“CatBus says HONK”);
     }
     /** Allows CatBug to honk at other CatBuses */
     public void conversation(CatBus target, int duration) {
          for(int i = 0; i < duration; i++) {
               honk();
               target.honk();
          }
     }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
public class CatBus _______________________________________ {

     _______________________ {
          System.out.println(“Purrrrrrr”);
     }

     __________________ {
          System.out.println(“CatBus says HONK”);
     }
     /** Allows CatBug to honk at other CatBuses */
     public void conversation(CatBus target, int duration) {
          for(int i = 0; i < duration; i++) {
               honk();
               target.honk();
          }
     }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
public class CatBus extends Cat ___________________________ {

     _______________________ {
          System.out.println("Purrrrrrr");
     }

     __________________ {
          System.out.println("CatBus says HONK");
     }
     /** Allows CatBug to honk at other CatBuses */
     public void conversation(CatBus target, int duration) {
          for(int i = 0; i < duration; i++) {
               honk();
               target.honk();
          }
     }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
public class CatBus extends Cat, implements Vehicle, Honker {

    _______________________ {
         System.out.println("Purrrrrrr");
    }

    __________________ {
         System.out.println("CatBus says HONK");
    }
    /** Allows CatBug to honk at other CatBuses */
    public void conversation(CatBus target, int duration) {
         for(int i = 0; i < duration; i++) {
              honk();
              target.honk();
         }
    }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
public class CatBus extends Cat, implements Vehicle, Honker {

     public void revEngine() {
          System.out.println("Purrrrrrr");
     }

     __________________ {
          System.out.println("CatBus says HONK");
     }
     /** Allows CatBug to honk at other CatBuses */
     public void conversation(CatBus target, int duration) {
          for(int i = 0; i < duration; i++) {
               honk();
               target.honk();
          }
     }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
public class CatBus extends Cat, implements Vehicle, Honker {

     public void revEngine() {
          System.out.println("Purrrrrrr");
     }

     public void honk() {
          System.out.println("CatBus says HONK");
     }
     /** Allows CatBug to honk at other CatBuses */
     public void conversation(CatBus target, int duration) {
          for(int i = 0; i < duration; i++) {
               honk();
               target.honk();
          }
     }
}
```

Fill in the CatBus class so CatBuses can rev their engines and honk at other CatBuses.

2A The Interfacing CatBus

```
/** Allows CatBug to honk at other CatBuses */
~~public void conversation(CatBus target, int duration) {~~
_________________________________________ {
    for(int i = 0; i < duration; i++) {
        honk();
        target.honk();
    }
}
```

Update the conversation method so that CatBuses can honk at CatBuses *and* Gooses.

2A The Interfacing CatBus

```
/** Allows CatBug to honk at other CatBuses */
~~public void conversation(CatBus target, int duration) {~~
public void conversation(Honker target, int duration) {
     for(int i = 0; i < duration; i++) {
          honk();
          target.honk();
     }
}
```

Update the conversation method so that CatBuses can honk at CatBuses *and* Gooses.

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

```
(A)____________________________
(B)____________________________
(C)____________________________
(D)____________________________
(E)____________________________
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________
```

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);                (A)____________________________
    Animal a = new Cat(“Olivia Benson”, 3);              (B)____________________________
    a = new Dog(“Fido”, 7);                              (C)____________________________
    System.out.println(a.greet());                       (D)____________________________
    a.playFetch();                                       (E)____________________________
    Dog d1 = a;                                          (F)____________________________
    Dog d2 = (Dog) a;                                    (G)____________________________
    d2.playFetch();                                      (H)____________________________
    (Dog) a.playFetch();                                 (I)____________________________
    Animal imposter = new Cat(“Pedro”, 12);              (J)____________________________
    Dog fakeDog = (Dog) imposter;                        (K)____________________________
    Cat failImposter = new Cat(“Jimmy”, 21);             (L)____________________________
    Dog failDog = (Dog) failImposter;                    (M)____________________________
}
```

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal("Nyan Cat", 5);
    Animal a = new Cat("Olivia Benson", 3);
    a = new Dog("Fido", 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat("Pedro", 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat("Jimmy", 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B)____________________________
(C)____________________________
(D)____________________________
(E)____________________________
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type
a	Animal	Cat

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal("Nyan Cat", 5);
    Animal a = new Cat("Olivia Benson", 3);
    a = new Dog("Fido", 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat("Pedro", 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat("Jimmy", 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C)____________________________
(D)____________________________
(E)____________________________
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D)____________________________
(E)____________________________
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type
a	Animal	Dog

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E)____________________________
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type
a	Animal	Dog

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

```
(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F)____________________________
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________
```

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type
a	Animal	Dog

3 Raining Cats & Dogs

```java
public static void main(String[] args) {
    Cat nyan = new Animal("Nyan Cat", 5);
    Animal a = new Cat("Olivia Benson", 3);
    a = new Dog("Fido", 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat("Pedro", 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat("Jimmy", 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G)____________________________
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G) no error
(H)____________________________
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog

```
public static void main(String[] args) {
    Cat nyan = new Animal("Nyan Cat", 5);
    Animal a = new Cat("Olivia Benson", 3);
    a = new Dog("Fido", 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat("Pedro", 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat("Jimmy", 21);
    Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G) no error
(H) Fetch, Fido!
(I)____________________________
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

```
(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G) no error
(H) Fetch, Fido!
(I) compile time error
(J)____________________________
(K)____________________________
(L)____________________________
(M)____________________________
```

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog
imposter	Animal	Cat

```java
public static void main(String[] args) {
     Cat nyan = new Animal("Nyan Cat", 5);
     Animal a = new Cat("Olivia Benson", 3);
     a = new Dog("Fido", 7);
     System.out.println(a.greet());
     a.playFetch();
     Dog d1 = a;
     Dog d2 = (Dog) a;
     d2.playFetch();
     (Dog) a.playFetch();
     Animal imposter = new Cat("Pedro", 12);
     Dog fakeDog = (Dog) imposter;
     Cat failImposter = new Cat("Jimmy", 21);
     Dog failDog = (Dog) failImposter;
}
```

(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G) no error
(H) Fetch, Fido!
(I) compile time error
(J) no error
(K)______________________________
(L)______________________________
(M)______________________________

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog
imposter	Animal	Cat

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);
    Animal a = new Cat(“Olivia Benson”, 3);
    a = new Dog(“Fido”, 7);
    System.out.println(a.greet());
    a.playFetch();
    Dog d1 = a;
    Dog d2 = (Dog) a;
    d2.playFetch();
    (Dog) a.playFetch();
    Animal imposter = new Cat(“Pedro”, 12);
    Dog fakeDog = (Dog) imposter;
    Cat failImposter = new Cat(“Jimmy”, 21);
    Dog failDog = (Dog) failImposter;
}
```

```
(A) compile time error
(B) no error
(C) no error
(D) Fido: Woof!
(E) compile time error
(F) compile time error
(G) no error
(H) Fetch, Fido!
(I) compile time error
(J) no error
(K) runtime error
(L)____________________________
(M)____________________________
```

Fill in what is printed by each line and note any errors.

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog
imposter	Animal	Cat
failImposter	Cat	Cat

3 Raining Cats & Dogs

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);                  (A) compile time error
    Animal a = new Cat(“Olivia Benson”, 3);                (B) no error
    a = new Dog(“Fido”, 7);                                (C) no error
    System.out.println(a.greet());                         (D) Fido: Woof!
    a.playFetch();                                         (E) compile time error
    Dog d1 = a;                                            (F) compile time error
    Dog d2 = (Dog) a;                                      (G) no error
    d2.playFetch();                                        (H) Fetch, Fido!
    (Dog) a.playFetch();                                   (I) compile time error
    Animal imposter = new Cat(“Pedro”, 12);                (J) no error
    Dog fakeDog = (Dog) imposter;                          (K) runtime error
    Cat failImposter = new Cat(“Jimmy”, 21);               (L) no error
    Dog failDog = (Dog) failImposter;                      (M)____________________________
}
```

Fill in what is printed by each line and note any errors.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog
d2	Dog	Dog
imposter	Animal	Cat
failImposter	Cat	Cat

```
public static void main(String[] args) {
    Cat nyan = new Animal(“Nyan Cat”, 5);            (A) compile time error
    Animal a = new Cat(“Olivia Benson”, 3);          (B) no error
    a = new Dog(“Fido”, 7);                          (C) no error
    System.out.println(a.greet());                   (D) Fido: Woof!
    a.playFetch();                                   (E) compile time error
    Dog d1 = a;                                      (F) compile time error
    Dog d2 = (Dog) a;                                (G) no error
    d2.playFetch();                                  (H) Fetch, Fido!
    (Dog) a.playFetch();                             (I) compile time error
    Animal imposter = new Cat(“Pedro”, 12);          (J) no error
    Dog fakeDog = (Dog) imposter;                    (K) runtime error
    Cat failImposter = new Cat(“Jimmy”, 21);         (L) no error
    Dog failDog = (Dog) failImposter;                (M) compile time error
}
```

Fill in what is printed by each line and note any errors.

4 Back to ABC’s

```
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println(“A.h”); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println(“B.h”); }
}
```

```
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A();
C cb = new B();

aa.f(ab);
ab.f(aa);
bb.f(ab);
ab.f(bb);
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC's

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println("A.h"); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println("B.h"); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab);
ab.f(aa);
bb.f(ab);
ab.f(bb);
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC's

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println("A.h"); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println("B.h"); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa);
bb.f(ab);
ab.f(bb);
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC's

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println("A.h"); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println("B.h"); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab);
ab.f(bb);
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC's

```
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println("A.h"); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println("B.h"); }
}
```

```
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb);
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC’s

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println(“A.h”); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println(“B.h”); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb); // 3
bb.f(bb);
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC’s

```
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println(“A.h”); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println(“B.h”); }
}
```

```
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb); // 3
bb.f(bb); // 3
ab.h();
bb.h();
((A) bb).h();
```

4 Back to ABC’s

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println(“A.h”); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println(“B.h”); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb); // 3
bb.f(bb); // 3
ab.h(); // A.h
bb.h();
((A) bb).h();
```

4 Back to ABC’s

```
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println(“A.h”); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println(“B.h”); }
}
```

```
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb); // 3
bb.f(bb); // 3
ab.h(); // A.h
bb.h(); // B.h
((A) bb).h();
```

4 Back to ABC's

```java
class A {
    int x = 1;
    void f(A other) { System.out.println(x); }
    void f(B other) { System.out.println(x + 2); }
    static void h() { System.out.println("A.h"); }
}

class B extends A {
    int x = 2;
    void f(A other) { System.out.println(x); }
    static void h() { System.out.println("B.h"); }
}
```

```java
A aa = new A();
B bb = new B();
A ab = new B();
C ca = new A(); // CE
C cb = new B(); // CE

aa.f(ab); // 1
ab.f(aa); // 2
bb.f(ab); // 2
ab.f(bb); // 3
bb.f(bb); // 3
ab.h(); // A.h
bb.h(); // B.h
((A) bb).h(); // A.h
```

5 Flatten

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (_______________________________________) {
        _______________________________________
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (_______________________________________) {
        _______________________________________
        _______________________________________
        _______________________________________
        _______________________________________
    }
    return a;
}
```

5 Flatten

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int[] arr: x) { // First, we need to find out how big we need to make the array
        totalLength += arr.length;
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (______________________________________) {
        ______________________________________
        ______________________________________
        ______________________________________
        ______________________________________
    }
    return a;
}
```

5 Flatten

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int[] arr: x) {
        totalLength += arr.length;
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (int[] arr: x) { // Go through every array one more time
        ______________________________________
        ______________________________________
        ______________________________________
        ______________________________________
    }
    return a;
}
```

5 Flatten

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int[] arr: x) {
        totalLength += arr.length;
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (int[] arr: x) {
        for (int value: arr) { // Then through every value in each array
        ______________________________________
        ______________________________________
        }
    }
    return a;
}
```

5 Flatten

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int[] arr: x) {
        totalLength += arr.length;
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (int[] arr: x) {
        for (int value: arr) {
            a[aIndex] = value; // Put the value at aIndex
        ______________________________________
        }
    }
    return a;
}
```

5 Flatten

```java
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int[] arr: x) {
        totalLength += arr.length;
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (int[] arr: x) {
        for (int value: arr) {
            a[aIndex] = value;
            aIndex++; // Increment the aIndex
        }
    }
    return a;
}
```

attendance
bit.ly/abhi-attendance

feedback
bit.ly/abhi-feedback

slides: **bit.ly/abhi-disc**