Test 1 Review

Discussion 05

Announcements

- Test 1 on Wednesday, 02/16
- Enigma released!

Review

Fun with Methods

Method Overloading is done when there are multiple methods with the same name and return type, but different parameters.

```
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 is done when a subclass has a method with the exact same function signature as a method in its superclass.

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

Casting allows our compiler to overlook cases where we are calling a method that belongs to a subclass on a variable that is statically typed to be the superclass.

```
Animal a = new Dog();
Dog d = a;
Dog d = (Dog) a;
```

Dynamic Method Selection

Your computer. . .

@ Compile Time:

1. Check for valid variable assignments
2. Check for valid method calls (only considering static type)

@ Run Time:

1. Check for overridden methods
2. Ensure casted objects can be assigned to their variables (only considering dynamic type)

Fields are **always** chosen based on static type!

Worksheet

1A Classy Cats

```
1  public class Animal {
2      protected String name, noise;
3      protected int age;
4      public Animal(String name, int age) {
5          this.name = name;
6          this.age = age;
7          this.noise = “Huh?”;
8      }
9      public String makeNoise() {
10         if (age < 2) {
11             return noise.toUpperCase();
12         }
13         return noise;
14     }
15     public String greet() {
16         return name + “: “ + makeNoise();
17     }
18 }
```

```
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

```
1  public class Animal {
2      protected String name, noise;
3      protected int age;
4      public Animal(String name, int age) {
5          this.name = name;
6          this.age = age;
7          this.noise = “Huh?”;
8      }
9      public String makeNoise() {
10         if (age < 2) {
11             return noise.toUpperCase();
12         }
13         return noise;
14     }
15     public String greet() {
16         return name + “: “ + makeNoise();
17     }
18 }
```

```
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

```
1  public class Animal {
2      protected String name, noise;
3      protected int age;
4      public Animal(String name, int age) {
5          this.name = name;
6          this.age = age;
7          this.noise = “Huh?”;
8      }
9      public String makeNoise() {
10         if (age < 2) {
11             return noise.toUpperCase();
12         }
13         return noise;
14     }
15     public String greet() {
16         return name + “: “ + makeNoise();
17     }
18 }
```

```
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

```
1  public class Animal {
2      protected String name, noise;
3      protected int age;
4      public Animal(String name, int age) {
5          this.name = name;
6          this.age = age;
7          this.noise = “Huh?”;
8      }
9      public String makeNoise() {
10         if (age < 2) {
11             return noise.toUpperCase();
12         }
13         return noise;
14     }
15     public String greet() {
16         return name + “: “ + makeNoise();
17     }
18 }
```

```
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

```
1   public abstract class Animal {
2       protected String name;
3       protected String noise = “Huh?”;
4       protected int age;
5       public String makeNoise() {
6           if (age < 2) {
7               return noise.toUpperCase();
8           }
9           return noise;
10      }
11      public String greet() {
12          return name + “: “ + makeNoise();
13      }
14      public abstract void shout();
15      abstract void count(int x);
16  }
```

```
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

```
1  public abstract class Animal {
2      protected String name;
3      protected String noise = “Huh?”;
4      protected int age;
5      public String makeNoise() {
6          if (age < 2) {
7              return noise.toUpperCase();
8          }
9          return noise;
10     }
11     public String greet() {
12         return name + “: “ + makeNoise();
13     }
14     public abstract void shout();
15     abstract void count(int x);
16 }
```

```
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

```
1  public abstract class Animal {
2      protected String name;
3      protected String noise = “Huh?”;
4      protected int age;
5      public String makeNoise() {
6          if (age < 2) {
7              return noise.toUpperCase();
8          }
9          return noise;
10     }
11     public String greet() {
12         return name + “: “ + makeNoise();
13     }
14     public abstract void shout();
15     abstract void count(int x);
16 }
```

Make the Cat class compatible with Animal

```
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(...);
        }
    }
}
```

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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);          (A)_____________________________
3      Animal a = new Cat("Olivia Benson", 3);        (B)_____________________________
4      a = new Dog("Fido", 7);                        (C)_____________________________
5      System.out.println(a.greet());                 (D)_____________________________
6      a.playFetch();                                 (E)_____________________________
7      Dog d1 = a;                                    (F)_____________________________
8      Dog d2 = (Dog) a;                              (G)_____________________________
9      d2.playFetch();                                (H)_____________________________
10     (Dog) a.playFetch();                           (I)_____________________________
11     Animal imposter = new Cat("Pedro", 12);        (J)_____________________________
12     Dog fakeDog = (Dog) imposter;                  (K)_____________________________
13     Cat failImposter = new Cat("Jimmy", 21);       (L)_____________________________
14     Dog failDog = (Dog) failImposter;              (M)_____________________________
15 }
```

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

variable	static type	dynamic type

3 Raining Cats & Dogs

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Cat

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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.

3 Raining Cats & Dogs

variable	static type	dynamic type
a	Animal	Dog

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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.

3 Raining Cats & Dogs

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

```
1  public static void main(String[] args) {
2      Cat nyan = new Animal("Nyan Cat", 5);
3      Animal a = new Cat("Olivia Benson", 3);
4      a = new Dog("Fido", 7);
5      System.out.println(a.greet());
6      a.playFetch();
7      Dog d1 = a;
8      Dog d2 = (Dog) a;
9      d2.playFetch();
10     (Dog) a.playFetch();
11     Animal imposter = new Cat("Pedro", 12);
12     Dog fakeDog = (Dog) imposter;
13     Cat failImposter = new Cat("Jimmy", 21);
14     Dog failDog = (Dog) failImposter;
15 }
```

(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) no error
(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

```
1   public static void main(String[] args) {
2       Cat nyan = new Animal("Nyan Cat", 5);
3       Animal a = new Cat("Olivia Benson", 3);
4       a = new Dog("Fido", 7);
5       System.out.println(a.greet());
6       a.playFetch();
7       Dog d1 = a;
8       Dog d2 = (Dog) a;
9       d2.playFetch();
10      (Dog) a.playFetch();
11      Animal imposter = new Cat("Pedro", 12);
12      Dog fakeDog = (Dog) imposter;
13      Cat failImposter = new Cat("Jimmy", 21);
14      Dog failDog = (Dog) failImposter;
15  }
```

(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) no error
(M) compile time error

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

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A();
5   C cb = new B();
6
7   aa.f(ab);
8   ab.f(aa);
9   bb.f(ab);
10  ab.f(bb);
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab);
8   ab.f(aa);
9   bb.f(ab);
10  ab.f(bb);
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa);
9   bb.f(ab);
10  ab.f(bb);
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab);
10  ab.f(bb);
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb);
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb); // 3
11  bb.f(bb);
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb); // 3
11  bb.f(bb); // 3
12  ab.h();
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb); // 3
11  bb.f(bb); // 3
12  ab.h(); // A.h
13  bb.h();
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb); // 3
11  bb.f(bb); // 3
12  ab.h(); // A.h
13  bb.h(); // B.h
14  ((A) bb).h();
```

4 Back to ABC's

```
1   class A {
2       int x = 1;
3       void f(A other) { System.out.println(x); }
4       void f(B other) { System.out.println(x + 2); }
5       static void h() { System.out.println("A.h"); }
6   }
7
8   class B extends A {
9       int x = 2;
10      void f(A other) { System.out.println(x); }
11      static void h() { System.out.println("B.h"); }
12  }
```

```
1   A aa = new A();
2   B bb = new B();
3   A ab = new B();
4   C ca = new A(); // CE
5   C cb = new B(); // CE
6
7   aa.f(ab); // 1
8   ab.f(aa); // 2
9   bb.f(ab); // 2
10  ab.f(bb); // 3
11  bb.f(bb); // 3
12  ab.h(); // A.h
13  bb.h(); // B.h
14  ((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

```
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;
}
```