class Animal {

public void animalSound() {

System.out.println("The animal makes a sound");

}

}

class Pig extends Animal {

@Override

public void animalSound() {

System.out.println("The pig says: wee wee");

}

}

class Dog extends Animal {

@Override

public void animalSound() {

System.out.println("The dog says: bow wow");

}

}

Animal myAnimal = new Animal();

Animal myPig = new Pig();

Animal myDog = new Dog();

myAnimal.animalSound(); // Output: The animal makes a sound

myPig.animalSound(); // Output: The pig says: wee wee

myDog.animalSound(); // Output: The dog says: bow wow

class Calculator {

int add(int a, int b) { return a + b; }

double add(double a, double b) { return a + b; }

}

def can\_vote(user\_id, item\_id):

# Query your database to check existing votes

return not db.exists(user\_id=user\_id, item\_id=item\_id)

def cast\_vote(user\_id, item\_id):

if can\_vote(user\_id, item\_id):

db.insert\_vote(user\_id, item\_id)

return "Vote cast successfully!"

else:

return "You have already voted for this item."

[

{

"id": "abc123",

"title": "Your favorite programming language?",

"options": [

{"id": "1", "text": "Python"},

{"id": "2", "text": "JavaScript"}

],

"is\_active": true,

"created\_at": "2025-07-25T17:00:00Z"

},

// ... more polls ...

]

# Pseudocode for retrieving active polls

def get\_active\_polls():

return Poll.objects.filter(is\_active=True)

const filter = { is\_closed: false };

const sort = { created\_at: -1 };

const { polls } = await chatClient.queryPolls(filter, sort);

{

"closed\_by": "admin\_user\_id",

"closed\_reason": "Poll duration ended",

"closed\_at": 1753397940

}