Remember our class hierarchy

Employee
  ▼
LegalSecretary
  ▼
Secretary
  ▼
Marketer
  ▼
Lawyer
  ▼
LegalSecretary

OK?

Employee e = new Lawyer();

OK?

Employee e = new Lawyer();
  e.getSalary();
  e.getVacationForm();

Yes. A Lawyer is an Employee
Employee e = new Lawyer();
e.getSalary();
e.getVacationForm();

Yes
  ▶ All Employees have a getSalary() and getVacationForm()
  ▶ Java calls Lawyer’s methods

Employee e = new Lawyer();
e.takeDictation();

No
  ▶ some Employees can do this
  ▶ not all

Employee e = new Lawyer();
e.sue();

does this mean:
  ▶ when we call a method all Employees have, we call the right one
  ▶ Using e, can only call methods common to all Employees
Employee e = new Lawyer();
e.sue();

No

- Not all employees can sue
- polymorphism means:
  - when we call a method all Employees have, we call the right one
- Using e, can only call methods common to all Employees

and of course we can

Employee team[] = new Employee[3];

team[0] = new Marketer();
team[1] = new Lawyer();
team[2] = new LegalSecretary();

for (int i=0; i<team.length; i++)
    System.out.println(team[i].getSalary());