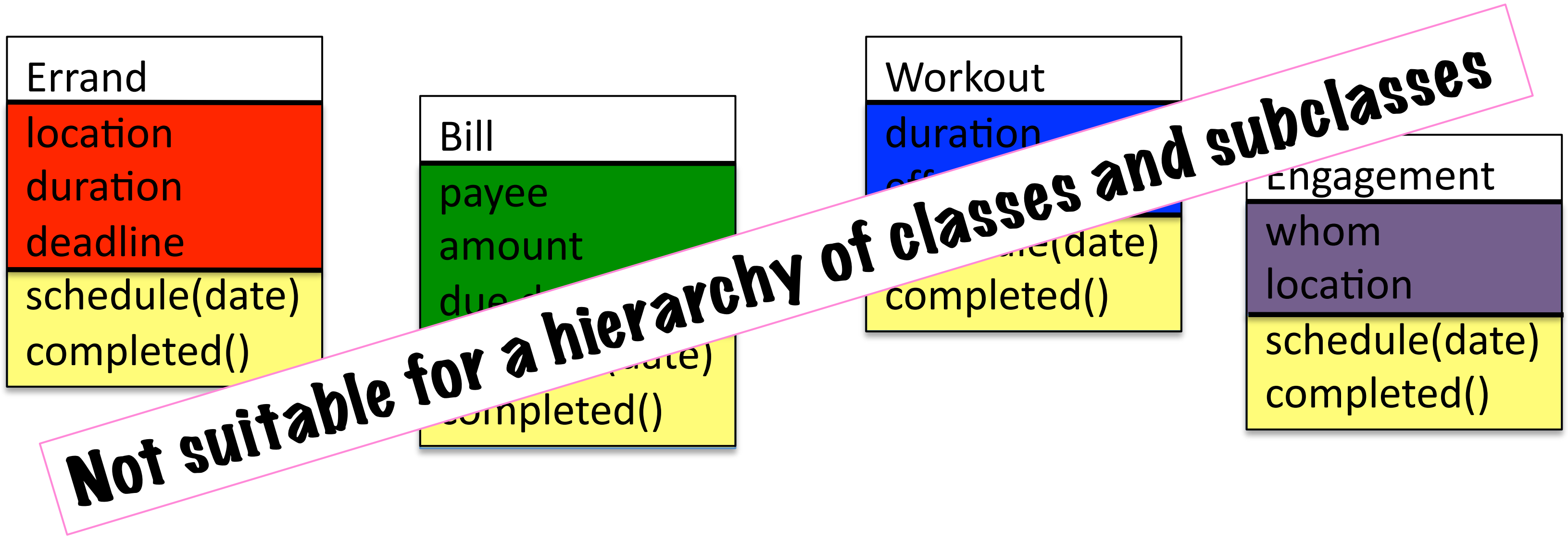
Look at examples

Seemingly unrelated objects in a To Do list app



Behavior of To Do list objects

Errand

location
duration
deadline

schedule(date)
completed()

Schedule an Errand:

✓ When must it be completed by?

```
dryCleaning.schedule(date);
```

✓ Find a time slot between now and then where it would fit

Engagement

whom
location

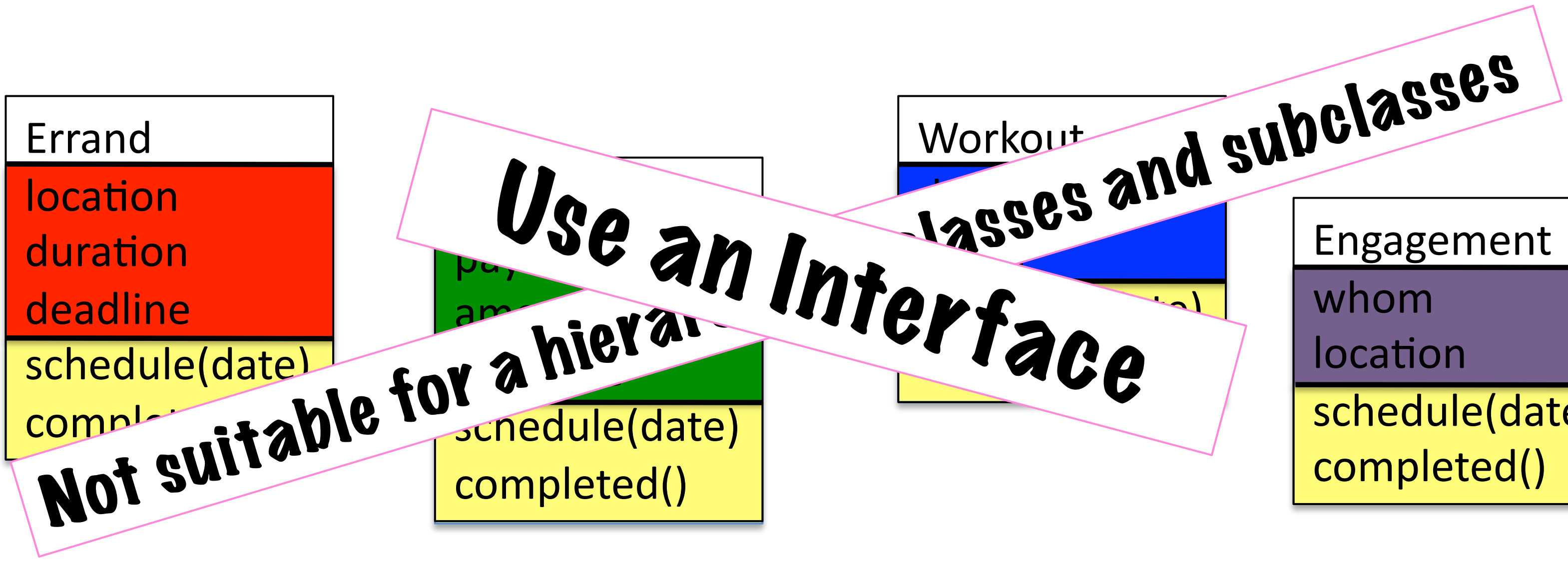
schedule(date)
completed()

Schedule an Engagement:

```
movieNight.schedule(date);
```

✓ Generate an R.S.V.P

Seemingly unrelated objects in a To Do list app



Java Interface

- Contains public method headings
  - return type
  - parameter list
  - no method body

```
//marks object completed on calendar
public void completed();
```

Java Interface

- File name should begin with a capital letter
- Is stored in a .java file
- No
  - constructors
  - instance variables
  - method bodies

Java Interface

```
public interface Item {
    // schedules item on given date
    //returns true if scheduled successfully
    public boolean schedule(Date day);

    //marks event as completed
    public void completed();

    other public methods...
}
```

A class that implements the interface

```
public class Errand implements Item{
    private String location;
    private int duration;
    private Date deadline;

    define constructors
    define necessary accessors and mutators

    public boolean schedule(Date day){
        if (day.before(this.deadline) ){
            code to mark calendar
            return (true);
        }
        else return (false);
    }

    public void completed(){
        code to define complete() method
}
```

A different class that implements the same interface

```
public class Engagement implements Item{
    private String whom;
    private String location;

    define constructors
    define necessary accessors and mutators

    public boolean schedule(Date day){
        if (day.available()){
            code to mark calendar
            sendRSVP(whom, true);
            return (true);
        }
        else return (false);
    }

    public void completed(){
        code to define complete() method
}
```

An interface as a parameter

- An interface can be a parameter type
- Polymorphism

An interface as a parameter

```
public boolean endDay(Item toDo, Date tomorrow){
   if (toDo.schedule(tomorrow)){
      toDo.completed();
      return (true);
   }
   else return (false);
}
```

every class that implements the interface `Item` **will have an implementation for** `schedule()` **and** `completed()`