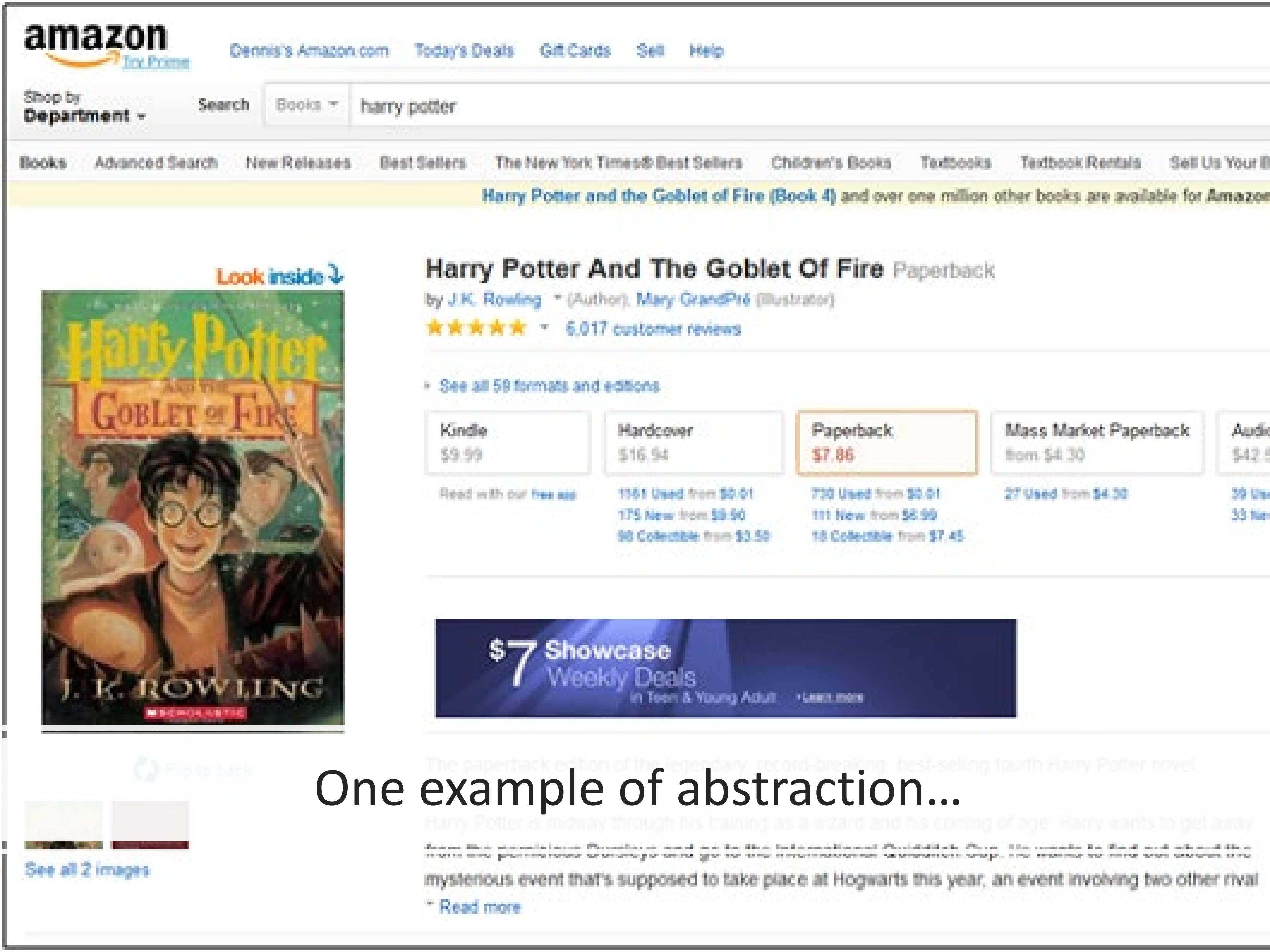
Class Design & Abstract Data Types

CIS110

Slides adapted from “Computational Thinking” by authors at Virginia Tech.

Class Design

- **Abstraction**: set of information properties relevant to a stakeholder about an entity
- Information Property (or property): a named, objective and quantifiable aspect of an entity
- Stakeholder: a real or imagined person (or a class of people) who is seen as the audience for, or user of the abstraction being defined



Harry Potter and the Goblet of Fire (Book 4) and over one million other books are available for Amazon

Harry Potter And The Goblet Of Fire Paperback

by J.K. Rowling (Author), Mary GrandPré (Illustrator)

★★★★★ 6,017 customer reviews

See all 59 formats and editions

Kindle	Hardcover	Paperback	Mass Market Paperback	Audi
$9.99	$16.94	$7.86	from $4.30	$42.5
Read with our free app	1161 Used from $0.01 175 New from $9.90 98 Collectible from $3.50	730 Used from $0.01 111 New from $6.99 18 Collectible from $7.45	27 Used from $4.30	39 Use 33 Ne

$7 Showcase Weekly Deals in Teen & Young Adult Learn more

The paperback edition of the legendary, record-breaking, best-selling fourth Harry Potter novel.

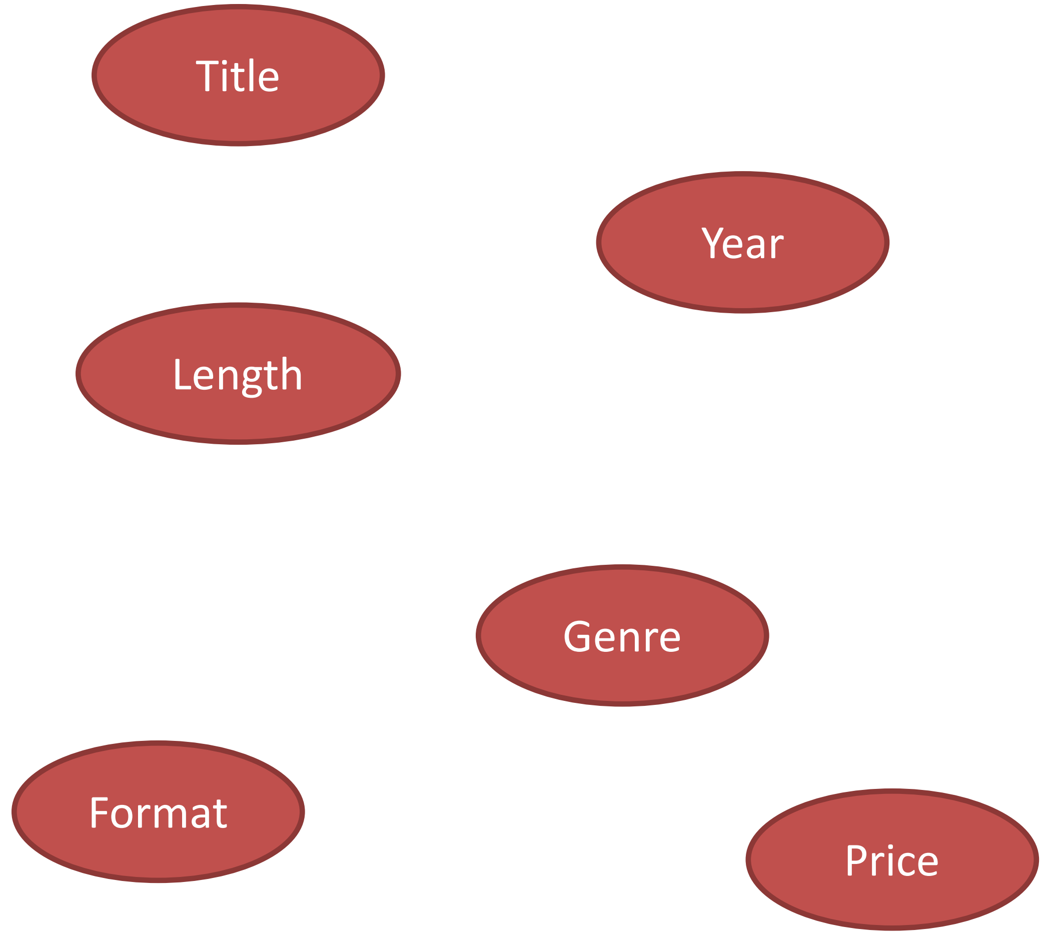
Harry Potter is midway through his training as a wizard and his coming of age. Harry wants to get away from the pernicious Dursleys and go to the International Quidditch Cup. He wants to find out about the mysterious event that's supposed to take place at Hogwarts this year, an event involving two other rival

Read more

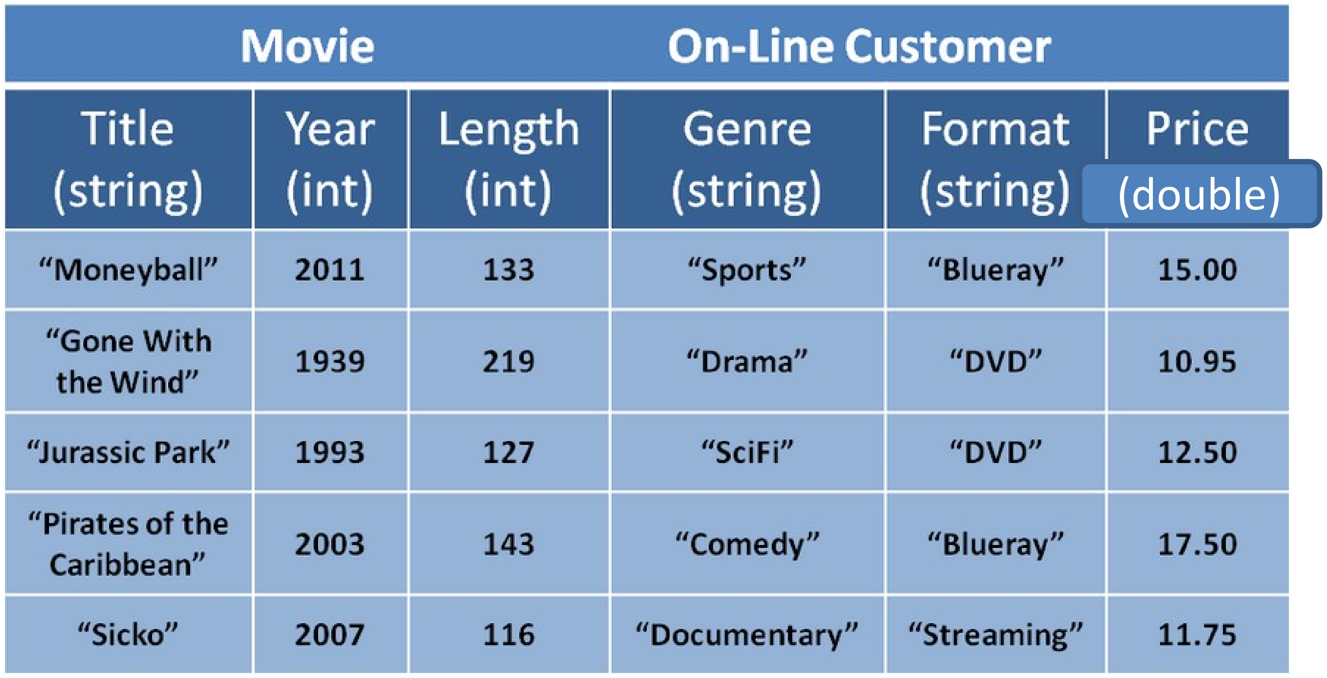
One example of abstraction...

Class Design

- Example:
  - Entity: Movie
  - Properties:



Class Design

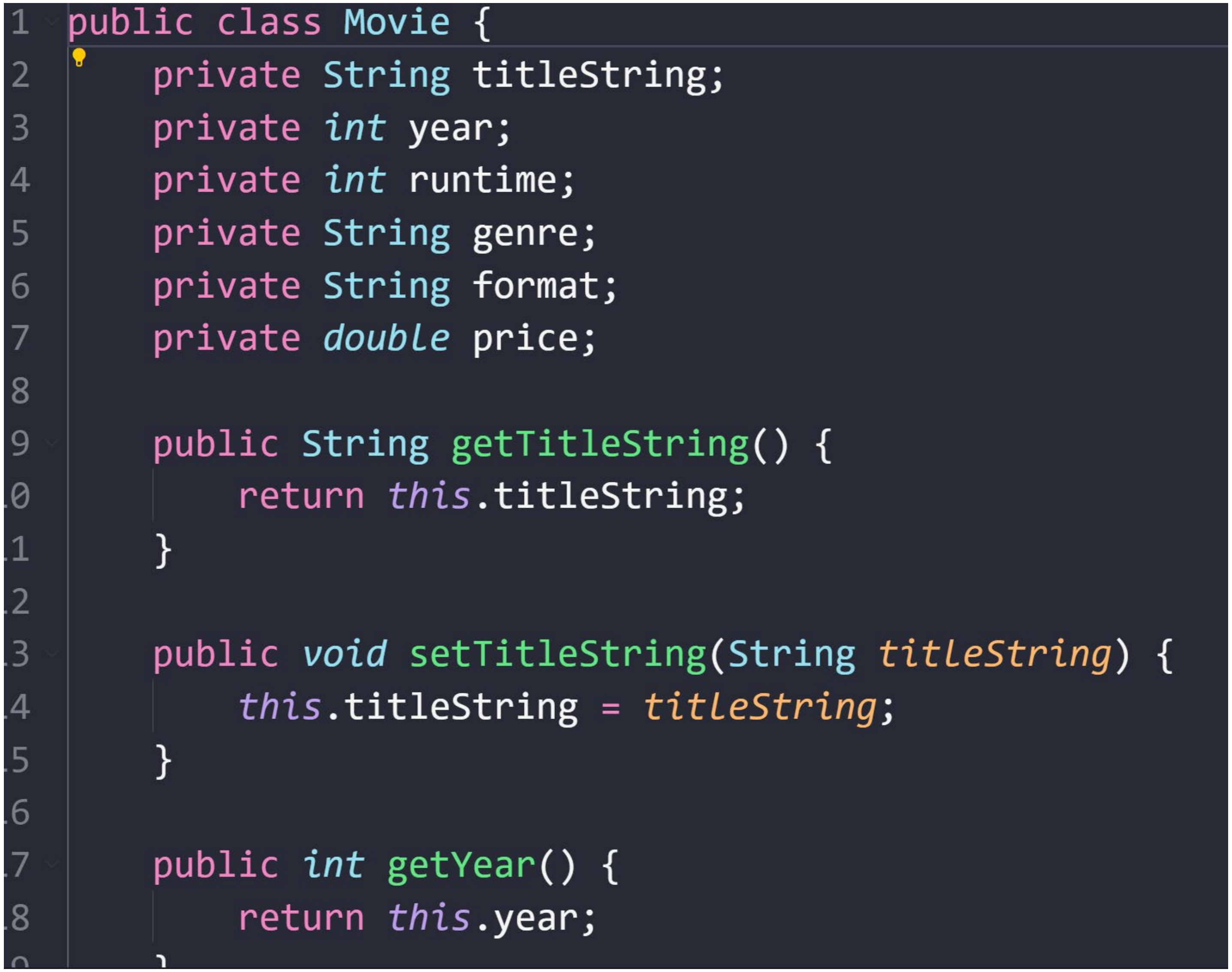


Movie			On-Line Customer		
Title (string)	Year (int)	Length (int)	Genre (string)	Format (string)	Price (double)
“Moneyball”	2011	133	“Sports”	“Blueray”	15.00
“Gone With the Wind”	1939	219	“Drama”	“DVD”	10.95
“Jurassic Park”	1993	127	“SciFi”	“DVD”	12.50
“Pirates of the Caribbean”	2003	143	“Comedy”	“Blueray”	17.50
“Sicko”	2007	116	“Documentary”	“Streaming”	11.75

Representing the Movie Abstraction using a Table

Java Class

- Entity is the class: Movie.Java
- Properties become attributes (data fields)
  - `String title`
  - `int year`
  - etc.



```java
public class Movie {
    private String titleString;
    private int year;
    private int runtime;
    private String genre;
    private String format;
    private double price;

    public String getTitleString() {
        return this.titleString;
    }

    public void setTitleString(String titleString) {
        this.titleString = titleString;
    }

    public int getYear() {
        return this.year;
    }
```

Start of Movie.java

Abstract Data Types (ADT)

Barbara Liskov

- First woman to get a Ph.D. in Computer Science in the USA (Stanford 1968)
- Turing Award, 2008
- Inventor of Abstract Data Types

Abstract Data Types

- An approach to computer representation of abstraction
- Only the use which may be made of an abstraction is relevant
- How the abstraction is implemented is irrelevant.
- Defines a class of abstract objects which is completely characterized by the operations (functions/methods) available on those objects.
- An abstract data type can be defined by defining the characterizing operations for that type

Using Abstract Data Types

1. An abstract object (an ADT is the object’s type) may be operated upon by the operations which define its abstract type
2. An abstract object may be passed as a parameter to a procedure (function/method)
3. An abstract object may be assigned to a variable, but only if the variable is declared to hold objects of that type

ADT in Java: `interfaces`

- An interface
  - Defines an ADT in Java
  - An interface is a *class-like* construct that contains only constants and abstract methods
  - An abstract method is a method that is not implemented. Only the method signature is listed
  - A constant is a variable which value does not change during the execution of the program. They are declared static and final
  - Gives a type for an object based on what it *does*, not on how it was implemented
  - Describes a contract that objects must satisfy

Defining an interface

```
public interface InterfaceName {
   constant declarations;
   abstract method signatures;
}
```

- Example:

```
public interface Shape {
   public static final double PI = 3.14159;
   public double area();
   public double perimeter();
   public void draw();
}
```

Implementing an interface

- Define a class that will implement the interface
- The class implementing the interface must implement all the methods defined in the interface
- The class implementing an interface declares a **subtype** of the interface
- The interface is a **supertype** of the implementation class
- A class can have multiple supertypes
- An interface can have multiple subtypes

```
public class Circle implements Shape
{    (truncated for space)

  private double radius;

  public Circle(double radius){
    this.radius = radius;
  }

  @Override
  public double area() {
    return radius * radius * PI;
  }
  …
}
```

Defined in the interface

Implementing an interface: `@Override`

- The `@Override` keyword indicates that the method implements/overrides a method defined in the interface
- Optional but very useful
- If the interface changes, methods “decorated” with `@Override` keyword will raise a compiler error. To fix the problem, make your code to adhere to the new interface

Using an interface

- Declare an object of type the interface and initialize it using the subtype constructor.
- Invoke the methods defined in the ADT on the object
- Example:

```
Shape c = new Circle(4);
c.area();
c.perimeter();
c.draw();
```