159 XTENDED

A WEEKLY REVIEW

INHERITANCE REVIEW

Pull up **Socrative.com**

and join room **XTEND159**

QUESTION: WHAT CONFUSES YOU THE MOST ABOUT CLASS INHERITANCE IN JAVA?

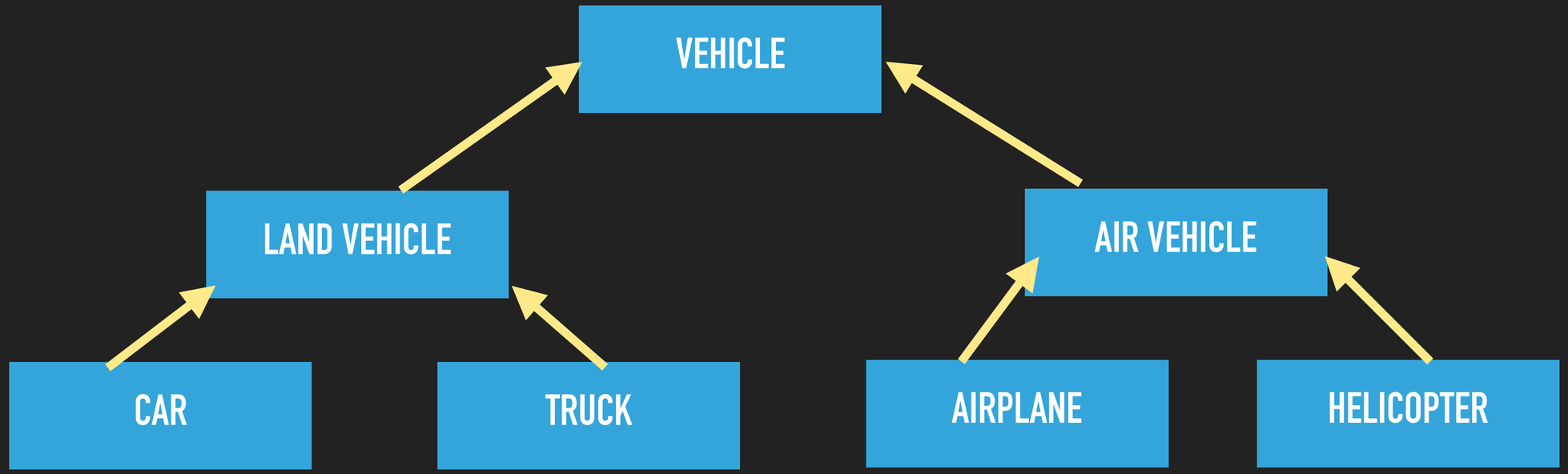
TRUE OR FALSE:
A CHILD CLASS THAT INHERITS FROM A SUPERCLASS CAN BE THE SUPERCLASS OF A DIFFERENT CLASS.

Pull up **Socrative.com**

and join room **XTEND159**

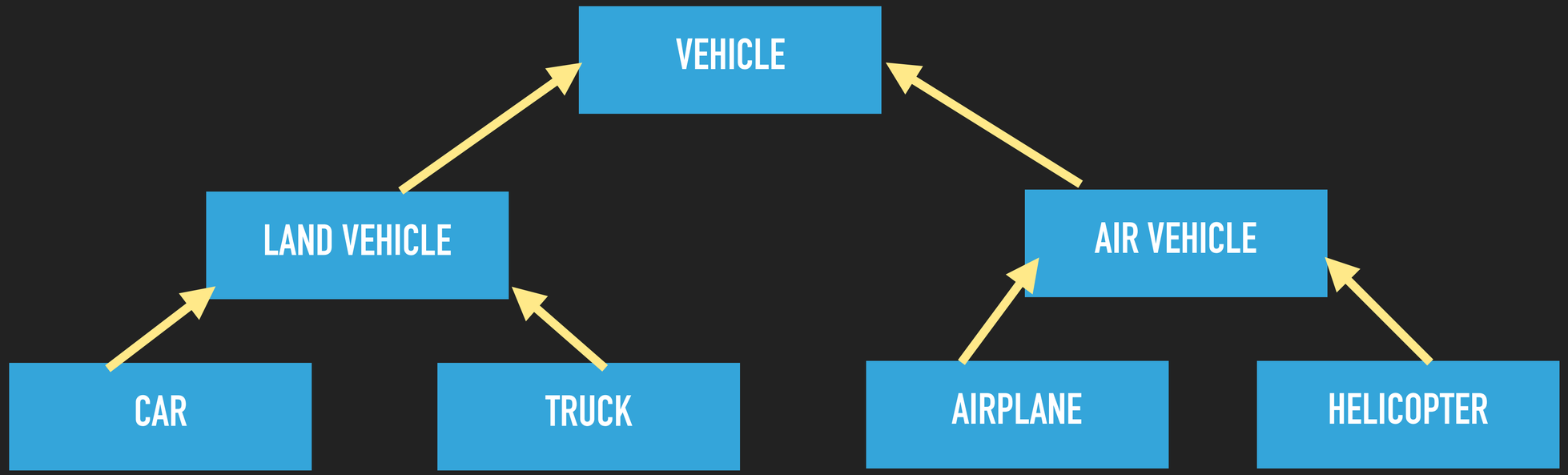
Answer individually first, then discuss as a table.

RELATIONSHIPS



- A Helicopter is an Air Vehicle. A Helicopter is not a Land Vehicle.
- A Helicopter is a Vehicle.
- An Air Vehicle may or may not be a helicopter.

RELATIONSHIPS

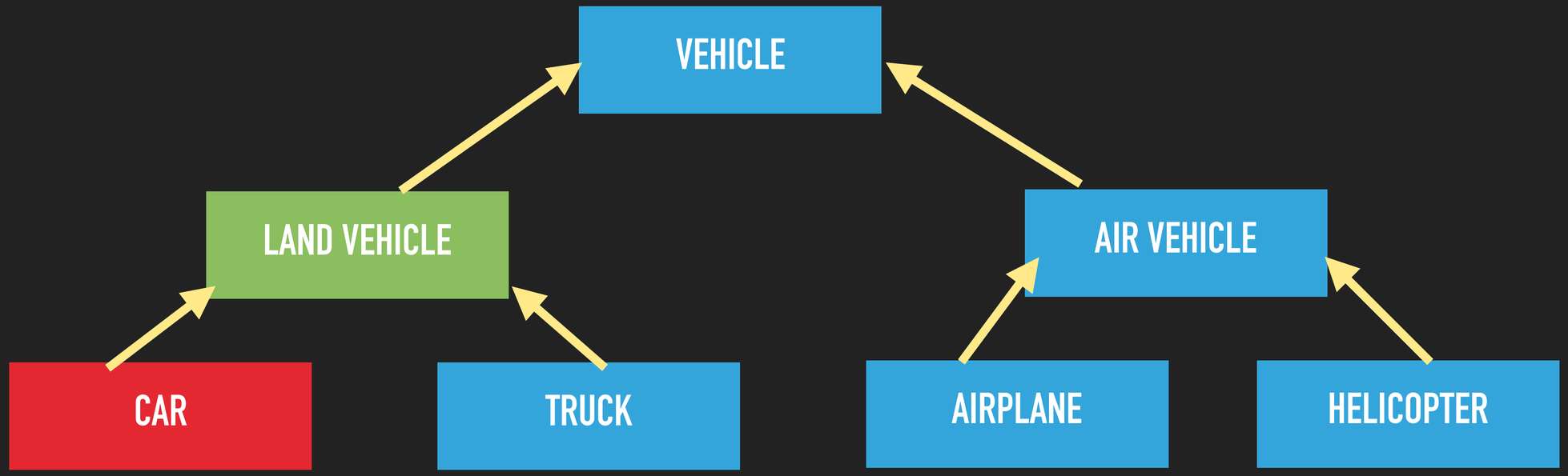


A Truck is a Land Vehicle.

More Specific
Less Generic

Less Specific
More Generic

RELATIONSHIPS



- A superclass is a more generalized class.
- A subclass is a more specialized class.

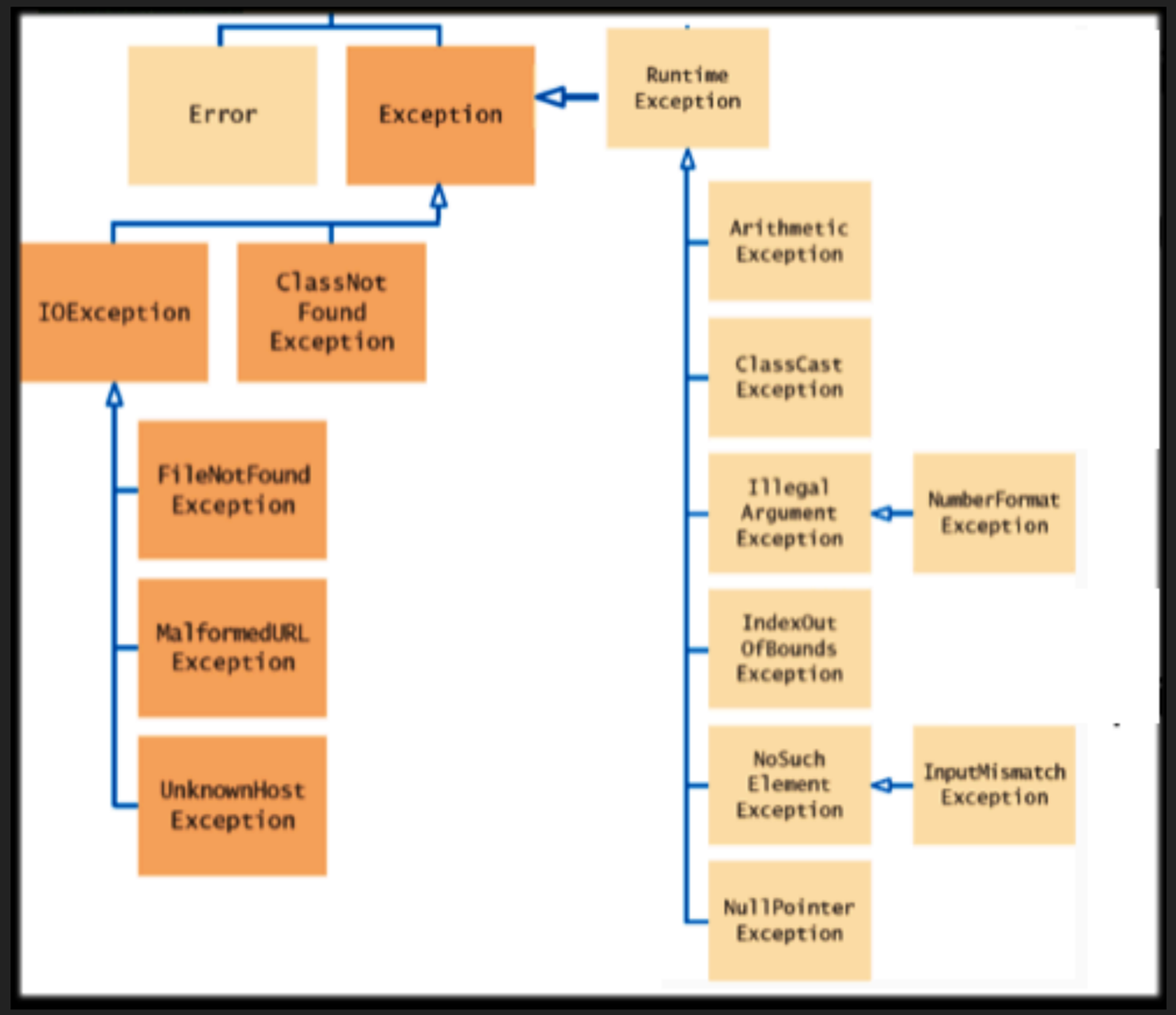
WHAT EXACTLY DOES A SUBCLASS INHERIT FROM A SUPERCLASS?

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

INHERITANCE EXAMPLE: EXCEPTIONS



- A “Runtime Exception” (also known as an Unchecked Exception) is a generic type of Exception.
- An ArithmeticException is a RuntimeException.
- A RuntimeException is an Exception.
- An IOException is an Exception.

TRUE OR FALSE:
A FILENOTFOUNDEXCEPTION IS A RUNTIME EXCEPTION.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

ACTIVITY: CARDS

```
public class Card
{
    // Instance Variables
    private String name;

    public Card(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return this.name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public boolean isExpired()
    {
        return false;
    }

    @Override
    public String toString()
    {
        return "Card Holder: " + name;
    }
}
```

- Write a class called `DebitCard` that will be a subclass of `Card`.
- A `DebitCard` will have two instance variables of type int: `cardNumber` and `pin`.
- Write a parameterized constructor that takes a `name`, `cardNumber`, and `pin`. It should call the superclass constructor.
- `DebitCard` should have a `toString` method that prints in this format:

Card Number: 12345678 Card Holder: Reece Adkins

ACTIVITY: CARDS

```
public class Card
{
    // Instance Variables
    private String name;

    public Card(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return this.name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public boolean isExpired()
    {
        return false;
    }

    @Override
    public String toString()
    {
        return "Card Holder: " + name;
    }
}
```

```
public class DebitCard extends Card
{
    // Instance Variables
    private int cardNumber;
    private int pin;

    public DebitCard(String name, int cardNumber, int pin)
    {
        super(name);
        this.cardNumber = cardNumber;
        this.pin = pin;
    }

    // Getter for cardNumber omitted
    // Setter for pin omitted

    @Override
    public String toString()
    {
        return "Card Number: " + cardNumber + super.toString();
    }
}
```

WHERE IS A PRIVATE VARIABLE VISIBLE?
WHERE IS A PROTECTED VARIABLE VISIBLE?

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

TRUE OR FALSE:
A CLASS CAN EXTEND TWO DIFFERENT CLASSES IN JAVA DIRECTLY.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

ACTIVITY: CARDS

```
public class Card
{
    // Instance Variables
    private String name;

    public Card(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return this.name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public boolean isExpired()
    {
        return false;
    }

    @Override
    public String toString()
    {
        return "Card Holder: " + name;
    }
}
```

- Write a class called `IDCard` that will be a subclass of `Card`.
- An `IDCard` will have two instance variables of type int: an `idNumber` and an `expirationYear`.
- Write a parameterized constructor that accepts a `name`, an `idNumber`, and an `expirationYear`.
- Write a method called `isExpired` which returns false if `expirationYear` is greater than or equal to 2020. It should return true otherwise.

ACTIVITY: CARDS

```
public class Card
{
    // Instance Variables
    private String name;

    public Card(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return this.name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public boolean isExpired()
    {
        return false;
    }

    @Override
    public String toString()
    {
        return “Card Holder: ” + name;
    }
}
```

```
public class IDCard extends Card
{
    // Instance Variables
    private int idNumber;
    private int expirationYear;

    public IDCard(String name, int idNumber, int expirationYear)
    {
        super(name);
        this.idNumber = idNumber;
        this.expirationYear = expirationYear;
    }

    // Getters and setters omitted

    @Override
    public boolean isExpired()
    {
        return (expirationYear < 2020);
    }
}
```

OVERRIDING METHODS

- A subclass can **override** a method from a superclass.

```
IDCard studentCard = new IDCard("Reece Adkins", 10011001, 2022);
if ( studentCard.isExpired() )
    System.out.println("Your card is expired.");
```

- From our previous activity, this statement calls the `isExpired` method from the subclass `IDCard`, not the superclass `Card`.
- How can we call the superclass implementation of `isExpired` from our subclass? Using `super.isExpired();`

WHAT IS THE DIFFERENCE BETWEEN METHOD OVERRIDING AND METHOD OVERLOADING?

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

TRUE OR FALSE:
THE CLASS "OBJECT" IS THE SUPERCLASS OF ALL CLASSES IN JAVA.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

TRUE OR FALSE:
ALL CLASSES CONTAIN A "COMPARETO" METHOD, WHETHER OR NOT YOU ADDED IT.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

159 XTENDED

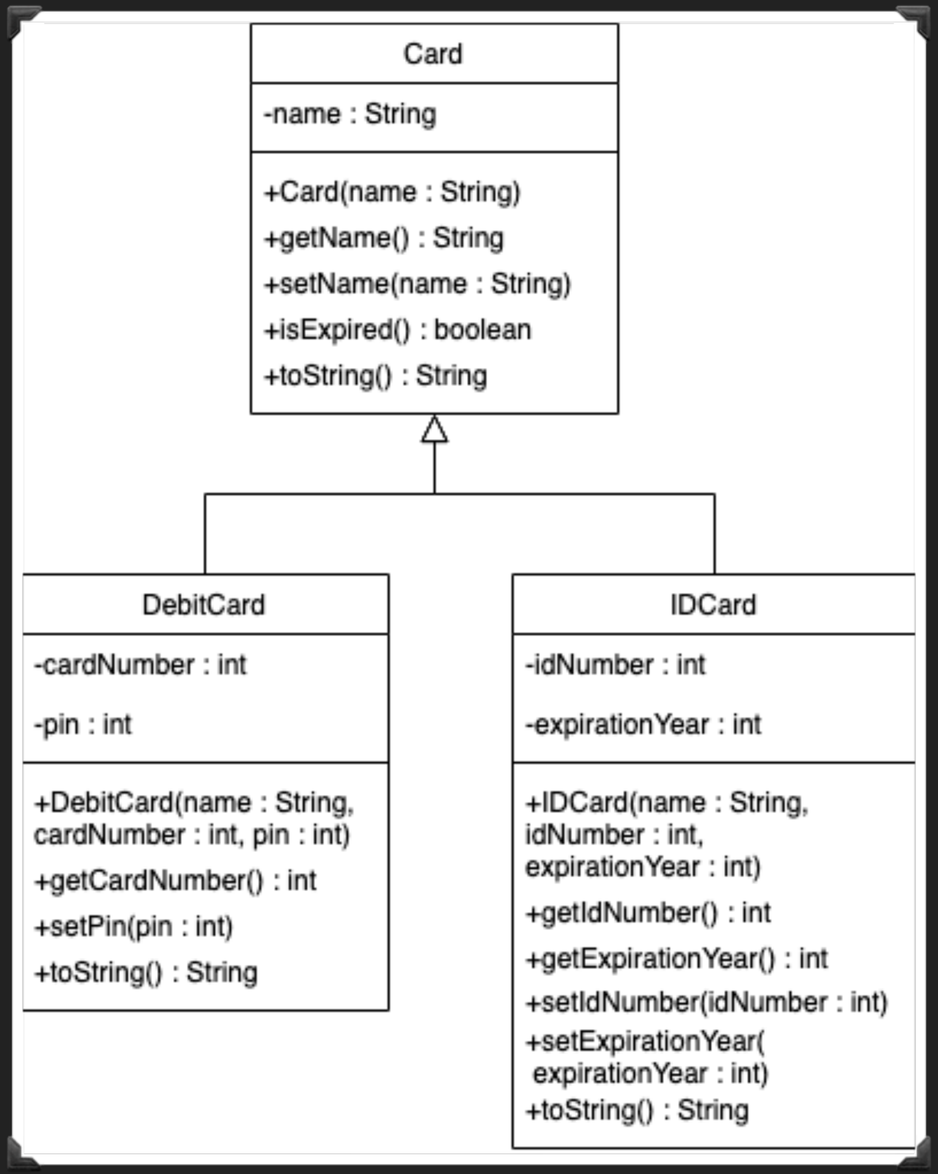
A WEEKLY REVIEW

POLYMORPHISM REVIEW

Reece Adkins, Cindy Zastudil, Zeru Tadesse, Becky Woods

POLYMORPHIC CARD EXAMPLE

- Write a method called printName that will accept an `IDCard` or a `DebitCard` and print the name of the cardholder.
- Create an `IDCard` object and pass it to `printName`, then show the output.



POLYMORPHIC CARD EXAMPLE

```
public void printName(Card aCard)
{
    System.out.print(aCard.getName());
}

IDCard studentCard = new IDCard("Reece Adkins", 10011001, 2022);
DebitCard bankCard = new DebitCard("Cindy Zastudil", 123456789, 1111);

printName(studentCard);
printName(bankCard);
```

Reece Adkins
Cindy Zastudil

POLYMORPHIC CARD EXAMPLE

```
public void printName(Card aCard)
{
    System.out.print(aCard.getName());
}

IDCard studentCard = new IDCard("Reece Adkins", 10011001, 2022);
DebitCard bankCard = new DebitCard("Cindy Zastudil", 123456789, 1111);

printName(studentCard);
printName(bankCard);
```

```
public void printName(Card aCard)
{
    System.out.print(aCard.getName());
}

Card studentCard = new IDCard("Reece Adkins", 10011001, 2022);
Card bankCard = new DebitCard("Cindy Zastudil", 123456789, 1111);

printName(studentCard);
printName(bankCard);
```

Question:

What has changed in this code?

Is it's functionality different after the change?

POLYMORPHIC CARD EXAMPLE

```
public void printName(Card aCard)
{
    System.out.print(aCard.getName());
}

Card studentCard = new IDCard("Reece Adkins", 10011001, 2022);
Card bankCard = new DebitCard("Cindy Zastudil", 123456789, 1111);

printName(studentCard);
printName(bankCard);
```

A Card variable can hold an instance of one of it's subclasses.
This allows all inherited attributes of Card inside of an IDCard and DebitCard to be accessed directly through a Card variable.

TRUE OR FALSE:
AN INSTANCE OF ANY CLASS IN JAVA CAN BE STORED IN AN OBJECT VARIABLE.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.

TRUE OR FALSE:
A CHILD CLASS VARIABLE CAN BE USED TO REFER TO AN INSTANCE OF IT'S SUPERCLASS.

Pull up **Socrative.com**

and join room **XTEND159**

Answer individually first, then discuss as a table.