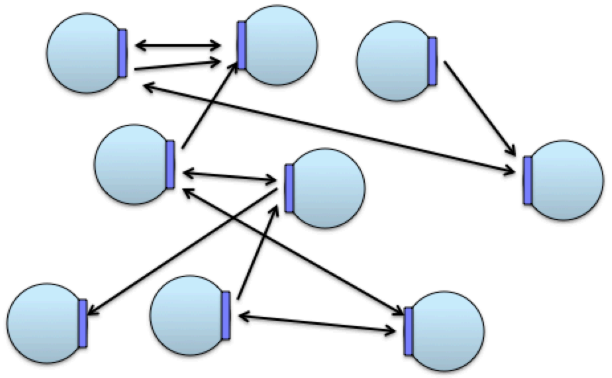
Lab 04: Java Interfaces

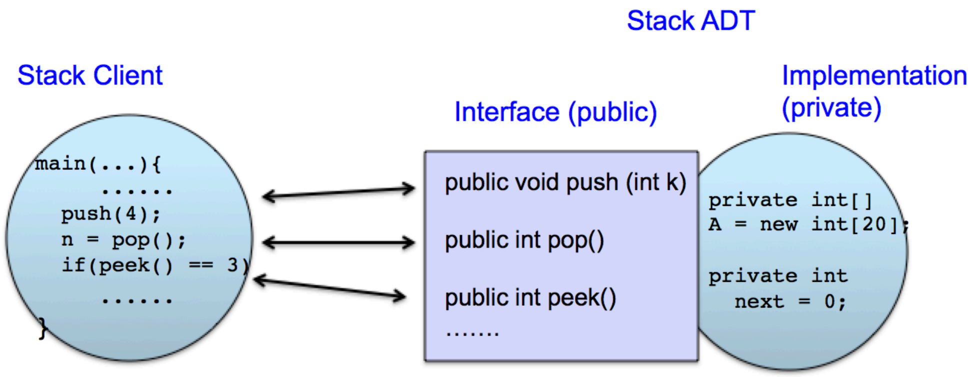
Computer Science

Recall: The way we control communication in Object-Oriented Design is through the **interface** of a class:



Interface = collection of public methods and fields of a class

The interface defines the (public) behavior of a class, which is separated from the (private) implementation:



```
main(...){
   ......
  push(4);
  n = pop();
  if(peek() == 3)
   ......
}
```

```
private int[]
A = new int[20];

private int
  next = 0;
```

Lab 04: Java Interfaces

Client.java

```
public class Client {
   public static void main(String [] args) {
      Collection C = new Collection();
      C.insert(2);
      C.insert(3);
      C.delete(2)
      if(C.member(2))
           System.out.println("Oh no....");
   }
}
```

Collection.java

```
public class Collection {
   private int [] A = new int[10];
   private int next = 0;

   public void insert(int k) {
        A[next++] = k
   }

   public void delete(int k) {
           ...   etc.   .....
   }

   public boolean member(int k) {
        ..... etc.    .....
   }
}
```

Interface in Red

Implementation in Green

Lab 04: Java Interfaces

Computer Science

Client.java

```
public class Client {
   public static void main(String [] args) {
      Collectable C = new Collection();
      C.insert(2);
      C.insert(3);
      C.delete(2)
      if(C.member(
         System.o
   }
}
```

Rule 2: The client using the interface can ONLY use methods which are in the interface.

Collectable.java

```
public interface Collectable {
     public void insert(int k) ;
     public void delete(int k) ;
     public boolean member(int k) ;
}
```

Collection.java

```
public class Collection implements Collectable
{
   private int [] A = new int[10];
   private int next = 0;

   public void insert(int
       A[next++] = k
   }

   public void delete(int
        ...   etc.   .....
   }

   public boolean member(int k) {
       ..... etc.    .....
   }
}
```

Rule 1: The ADT class implementing the interface must provide **implementations** for all the methods in the interface. Can provide other public methods if it wants.

Lab 04: Java Interfaces

Computer Science

Client.java

```
public class Client {
  public static void main(String [] args) {
    Collectable C = new Collection();
    C.insert(2);
    C.insert(3);
    C.delete(2)
    if(C.member(2))
        System.out.println("Oh no....");
  }
}
```

Collectable.java

```
public interface Collectable {
    public void insert(int k) ;
    public void delete(int k) ;
    public boolean member(int k) ;
}
```

Collection.java

```
public class Collection implements Collectable
{
  private int [] A = new int[10];
  private int next = 0;

  public void insert(int k) {
      A[next++] = k
  }

  public void delete(int k) {
        ...   etc.   .....
  }

  public boolean member(int k) {
      ..... etc.    .....
  }
}
```

Lab 04: Java Interfaces

Think of an **interface** as a **contract** between the Client and the ADT:

Client: "I need **insert**, **delete**, and **member** methods."

ADT: "No problem."

Client: "Wait, I just met you. How can I **trust** you?"

ADT: "We'll let Java check that the **contract** covers all your needs and that I provide everything in the contract"

Client: "What if I don't need everything in the contract? What if you offer more?"

ADT: "What do you care! As long as you get what you contracted for, you can run!"

Client: "You arrogant tech guys are all alike.... Ok, whatever, where do I sign?"

In mathematical terms, if C is what the client needs, F is what is listed in the interface, and D is what the ADT provides, we have: $C \leq F \leq D$.