

Assignment Project Exam Help
Collections and
<https://tutorcs.com>
Generics
WeChat: cstutorcs

Objectives

- Define the terms collection and ADT
- Introduce the types of collections that will be covered in this course
- Define the concept of an interface
- Identify when and how to use generics

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Collections

Collection: a group of items that we wish to treat as a conceptual unit.

Assignment Project Exam Help

- The proper choice of a collection for a given problem can improve the efficiency and simplicity of a solution.

<https://tutorcs.com>

WeChat: cstutorcs

Abstract Data Type (ADT)

It is a **collection** of data together with the **operations** on that data.

Assignment Project Exam Help

Each ADT is a specific model that defines which methods belong to the collection.

<https://tutorcs.com>
WeChat: cstutorcs

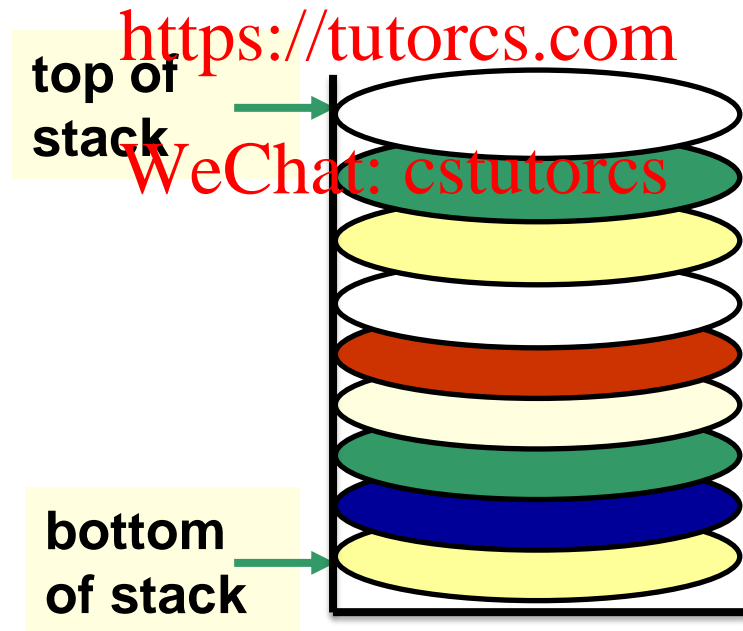
The ADT specifies **WHAT** the operations do, not **HOW** they do it.

ADT

- For example, there might be at least one operation to add elements to a collection.
[Assignment Project Exam Help](https://tutorcs.com)
 - Where does the new element get added?
<https://tutorcs.com>
 - Is it always added to the end? The front?
[WeChat: cstutorcs](#)
- In this course, we will learn about these ADTS: **stacks**, **queues**, **lists**, and **trees**.

Stacks

Stack: a collection whose elements are added and removed from one end, called the ***top*** of the stack.

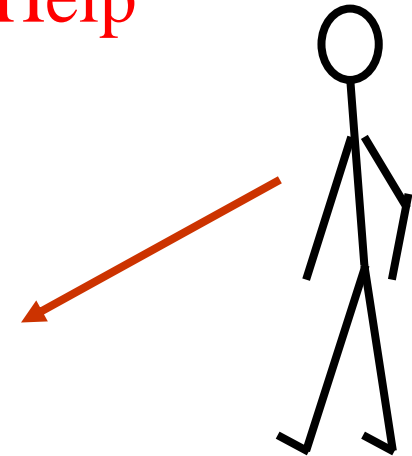


Queues

Queue: a collection whose elements are added to the rear and removed from the front.

Assignment Project Exam Help

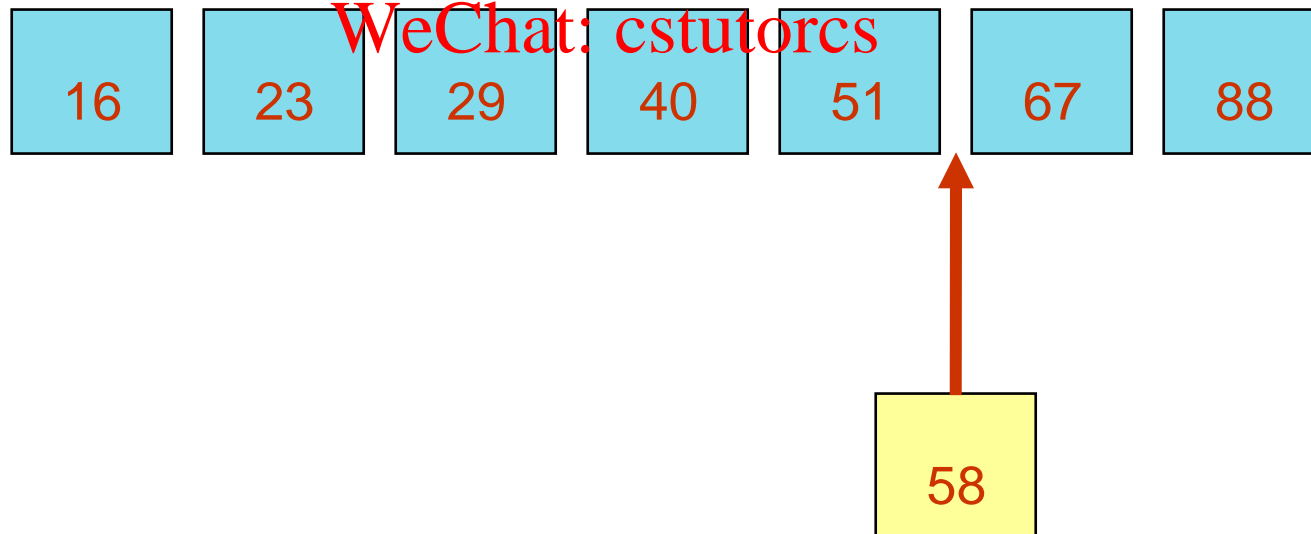
First person served
will be the one at
the front of queue



New person is
added to the rear
of the queue

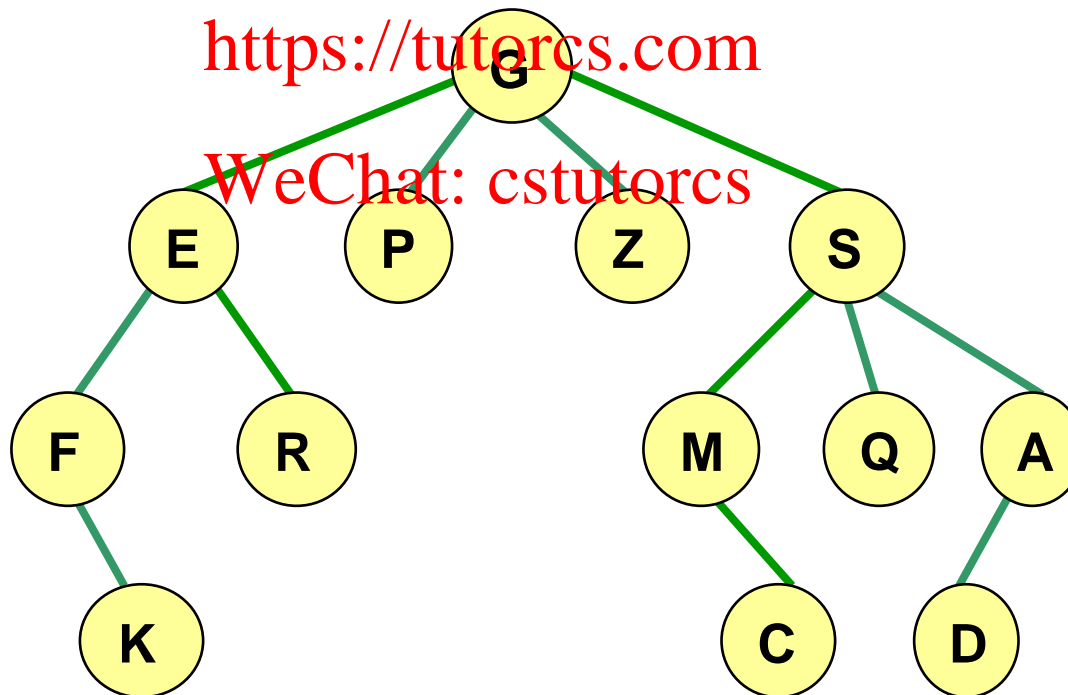
Lists

Lists: there are several types of lists; this is an example of an Ordered List. New numbers must be added such that all the numbers remain in order.



Trees

Trees: a non-linear collection whose elements are represented hierarchically with parent-child relationships.



Abstraction

- Abstraction separates the ***purpose*** of an entity from its ***implementation*** or how it works
 - ***Example in real life***: a car (we do not have to know how an engine works in order to drive a car)
 - ***Example in computer systems***: a computer (we do not need to know how information is stored and manipulated by the CPU to be able to execute programs)

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Java Interfaces

Java has a **programming construct** called an **interface** that we can use to define what the operations on an ADT are.

Assignment Project Exam Help

<https://tutorcs.com>

A Java interface is a list of **abstract methods** (the signatures of the methods) and constants

WeChat: cstutorcs

- A Java interface must be **public**
- Constants must be declared as **static final**



Java Interfaces

An interface is essentially a template for classes. Classes that implement an interface must contain **all** methods defined by the interface.

<https://tutorcs.com>

WeChat: cstutorcs

Interfaces cannot be instantiated, however they can be a *variable type*. i.e.

- `MyInterface data = new MyInterface();`  Invalid.
- `MyInterface data = new MyClass();`  Valid, if MyClass implements MyInterface.

A Java interface Specifies an ADT

```
public interface SimpleADT {  
    // Add a new element to the collection.  
    public void add ( String dataItem );  
    // Remove an element from the collection.  
    public void remove ( String dataItem );  
    // Returns an item at a specified position in the collection.  
    public String getItem ( int pos );  
    // Returns the number of elements in this collection.  
    public int size ( );  
    // Returns a string representation of this collection.  
    public String toString ( );  
}
```

Assignment Project Exam Help
<https://tutorcs.com>
WeChat: cstutorcs

Java interfaces Must be Implemented by Classes

```
public class SimpleCollection implements SimpleADT {  
    private String[] array; // Use an array to store items.  
    private int count; // Keep track of item count.  
  
    // What methods are required in here?  
  
    WeChat: cstutorcs  
  
}
```

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Generic Types

What data type(s) can we store in SimpleCollection or any other implementation of SimpleADT?

What if we wanted to store other types? Could we create a SimpleCollection to store Integers?

Doubles? Person objects? Rectangle objects?

This is where **generics** can help

- Generics allow us to make interfaces and classes that work for *any* data type.
- To do this, the interface/class definition needs a parameter; class parameters are enclosed between angle brackets: **<T>**, **<E>**, **<MyType>**, etc.
- It is conventional to use **<T>** but any letter or sequence of letters is allowed in the **< >**.

Generic Types

- Note that in generics we cannot use primitive types. However, there are **wrapper classes** for them, i.e. Integer for int, Double for double, Boolean for boolean, etc.

Assignment Project Exam Help

- The **actual type** is known only when an application program creates an object of that class
 - Example: <https://tutorcs.com>
WeChat: cstutorcs
 - SimpleCollection<String> s = new ...
 - SimpleCollection<Integer> n = new ...
 - SimpleCollection<Double> d = new ...
 - SimpleCollection<Person> p = new ...
 - SimpleCollection<Rectangle> r = new ...

Generic Types

- Will the class automatically handle other types just by adding <T>? *No!*

Assignment Project Exam Help

- We have to also adjust the methods a little to work with a generic type.

<https://tutorcs.com>

WeChat: cstutorcs

- We will have to use T anywhere the data type is being used. For example,

```
public void add ( String dataItem );
```

would be converted to:

```
public void add ( T dataItem );
```

A Java interface with String Parameters

```
public interface SimpleADT {  
    // Add a new element to the collection.  
    public void add ( String dataItem );  
    // Remove an element from the collection.  
    public void remove ( String dataItem );  
    // Returns an item at a specified position in the collection.  
    public String getItem ( int pos );  
    // Returns the number of elements in this collection.  
    public int size ( );  
    // Returns a string representation of this collection.  
    public String toString ( );  
}
```

Same Java interface with Generic Parameters

```
public interface SimpleADT<T> {  
    // Add a new element to the collection.  
    public void add ( T dataItem );  
    // Remove an element from the collection.  
    public void remove ( T dataItem );  
    // Returns an item at a specified position in the collection.  
    public T getItem ( int pos );  
    // Returns the number of elements in this collection.  
    public int size ( );  
    // Returns a string representation of this collection.  
    public String toString ( );  
}
```

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Generic Arrays

```
public class SimpleCollection<T> implements  
    SimpleADT<T> {  
    private T[] array; // Use an array to store items.  
    private int count; // Keep track of item count.
```

Assignment Project Exam Help

```
public SimpleCollection(int initCapacity) {  
    count = 0;  
    array = (T[])(new Object[initCapacity]);  
}
```



Why such complex declaration?

```
// Other methods...
```

```
}
```

Generic Arrays


```
public class SimpleCollection<T> implements  
    SimpleADT<T> {  
    private T[] array; // Use an array to store items.  
    private int count; // Keep track of item count.
```

Assignment Project Exam Help

```
    public SimpleCollection (int initCapacity) {  
        count = 0; WeChat: cstutorcs  
        array = new T[initCapacity];  
    }
```

<https://tutorcs.com>

WeChat: cstutorcs



Why is this wrong?

```
// Other methods...
```

```
}
```

Generic Arrays

```
public class SimpleCollection<T> implements  
    SimpleADT<T> {  
    private T[] array; // Use an array to store items.  
    private int count; // Keep track of item count.  
  
    public SimpleCollection (int initCapacity) {  
        count = 0;  
        array = new Object[initCapacity];  
    }  
  
    // Other methods...  
}
```

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Why is this wrong?

Data Structures

- Implementing collections is done with some underlying ***data structure***.
- The main two data structures used for this are: <https://tutorcs.com>
 - Arrays WeChat: cstutorcs
 - Linked Lists