



---

---

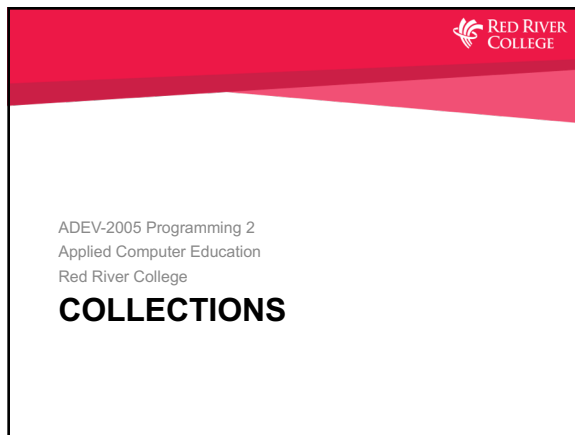
---

---

---

---

---



---

---

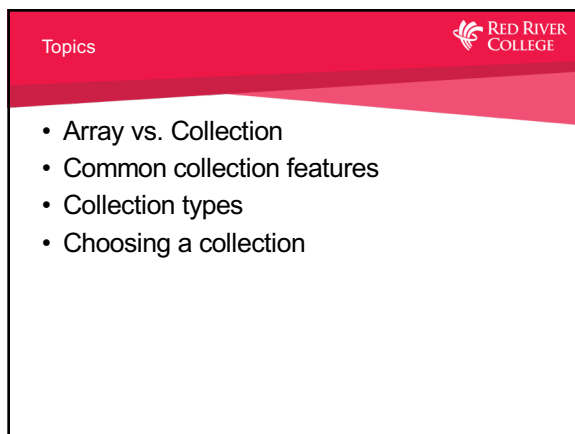
---

---

---

---

---



---

---

---


---

---

---

---

Array



- Zero indexed
- Single element type
- Fixed number of elements
- Fast

---

---

---


---

---

---

---

Array Example



```
int[] grades = new int[20];
grades[0] = 99;
grades[1] = 77;

foreach (int grade in grades)
{
    Console.WriteLine("{0}, ", grade);
}

decimal[] averages = { 12.3m, 23.4m, 34.5m, 45.6m, 56.7m, 67.8m, 78.9m, 89.0m };

for (int i = 0; i < averages.Length; i++)
{
    Console.WriteLine("{0}, ", averages[i]);
}
```

---

---

---


---

---

---

---

Collection Features



- Dynamic memory allocation
- Ability to be enumerated
  - Using foreach loop
- Ability to copy contents to an array
- Count, search and sort, etc.
- Zero indexed

---

---

---

---

---

---

---

## ArrayList



- System.Collections.ArrayList
- Like array, but can store various types
  - Heterogeneous collection
- Dynamic memory allocation
- Add/removing is slow

---

---

---

---

---

---

---

## ArrayList Example



```
ArrayList arrayList = new ArrayList();  
arrayList.Add("Bob");  
arrayList.Add(40);  
arrayList.Add(80000.00);  
  
arrayList.Insert(1, new HourlyEmployee("777", "Daniel", 60, 15.50m));  
arrayList.RemoveAt(2);  
Console.WriteLine(arrayList[1]);  
foreach (object item in arrayList)  
{  
    Console.Write("{0}, ", item);  
}
```

---

---

---

---

---

---

---

## List



- System.Collections.Generic.List
- Like an ArrayList...but single typed
  - Homogeneous collection
- Preferred over ArrayList
  - Heterogeneous (List<Object>)
- Dynamic memory allocation
- Values retrieved by index

---

---

---

---

---

---

---

## List Example



```
List<string> cities = new List<string>();

cities.Add("New York");
cities.Add("Mumbai");
cities.Add("Berlin");
cities.Add("Istanbul");

Console.WriteLine(cities[1]);

foreach (string item in cities)
{
    Console.Write("{0}, ", item);
}
```

---

---

---

---

---

---

---

## Dictionary (Associative Array)



- System.Collections.Generic.Dictionary
- Single type
- Key – Value Pair
- Elements retrieved by key

---

---

---

---

---

---

---

## Dictionary Example



```
Dictionary<string, Employee> dictionary =
    new Dictionary<string, Employee>();

dictionary.Add("222", new HourlyEmployee("222", "Dan", 50, 13));
dictionary.Add("777", new HourlyEmployee("777", "Jane", 50, 13));
dictionary.Add("101", new SalariedEmployee("101", "Beth", 34567));

Console.WriteLine(dictionary["222"]);

dictionary.Remove("222");

foreach (KeyValuePair<string, Employee> item in dictionary)
{
    Console.WriteLine("Key: {0}, Value: {1}", item.Key, item.Value);
}
```

---

---

---

---

---

---

---

## What collection to use choose?



- What data type or types are you storing?
- Do you know how many items?
  - Large data sets will require faster storage
- Ability to add/remove items?

---

---

---

---

---

---

---

## Summary



- Array = same as Java
- ArrayList = any type
- List = specific type
- Dictionary = any type, key/value pair
- Choose collection wisely

---

---

---

---

---

---

---

**QUESTIONS?**

---

---

---

---

---

---

---