Metropolitan State University, St. Paul, MN
ICS 372 Object-Oriented Design and Implementation
Class Exercise 4 Solution Part 1

1. Study following implementation of a bag and implement the `Iterator` interface.

```
import java.util.Iterator;
public class Bag implements Iterable {
   private int[] data;
   private int numberOfItems;

   public Bag(int capacity) {
      data = new int[capacity];
   }

   public void add(int item) {
      if (numberOfItems >= data.length) {
         allocateStorage((numberOfItems + 1) * 2);
      }
      data[numberOfItems++] = item;
   }

   public void allocateStorage(int newCapacity) {
      // Incomplete; don't implement in class
   }

   public boolean remove(int item) {
      // Incomplete; don't implement in class
      // remove item if present and return true
      // otherwise, return false
      return false;
   }

   @Override
   public Iterator iterator() {
      // TODO implement in class
      return null;
   }


}
```

```
import java.util.Iterator;
public class Bag implements Iterable {
   private int[] data;
   private int numberOfItems;
```

```
public Bag(int capacity) {
   data = new int[capacity];
}

public void add(int item) {
   if (numberOfItems >= data.length) {
      allocateStorage((numberOfItems + 1) * 2);
  }
  data[numberOfItems++] = item;
}

public void allocateStorage(int newCapacity) {
// Incomplete; don't implement in class
}

public boolean remove(int item) {
// Incomplete; don't implement in class
// remove item if present and return true
// otherwise, return false
   return false;
}

@Override
public Iterator iterator() {
   return new MyIterator();
}

private class MyIterator implements Iterator {
   int[] copy;
   int position;

   public MyIterator() {
      copy = new int[numberOfItems];
      position = 0;
      for (int index = 0; index < numberOfItems; index++) {
         copy[index] = data[index];
      }
   }

   @Override
   public boolean hasNext() {
      return position >= copy.length ? false : true;
   }

   @Override
   public Object next() {
      if (hasNext()) {
          return copy[position++];
```

```
            }
            return null;
        }
    }
```

2. Write code to use the above iterator and print all integers that are greater than 10 from the collection.

```
Bag aBag = new Bag(10);
aBag.add(8);
aBag.add(10);
aBag.add(8);
aBag.add(20);
aBag.add(30);
Iterator iterator = aBag.iterator();
while (iterator.hasNext()) {
   int value = (int) iterator.next();
   if (value > 10) {
      System.out.println(value);
   }
}
```

3. Implement a singleton class named `C` using the Java `static` block.

```
public class C {
   private static C cInstance;
   static {
      cInstance = new C();
   }

   private C() {
   }

   public static C instance() {
      return cInstance;
   }
}
```