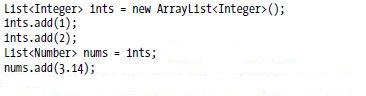
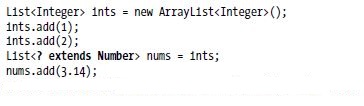
Lab 11

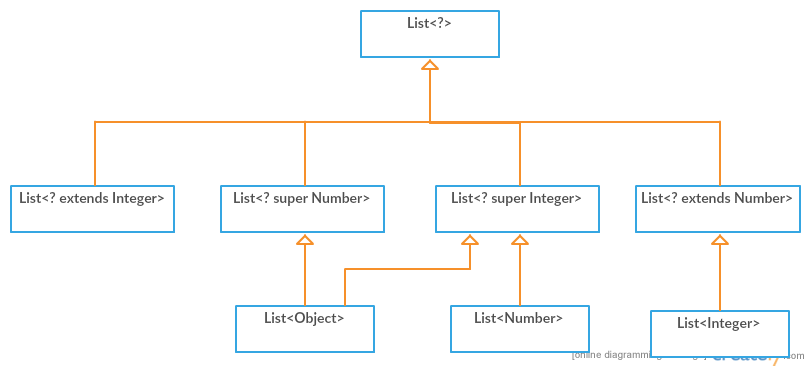
1. Consider the following code fragments. For each, if there is a compiler error, identify where it occurs.  
     
   a. First fragment:  
      
   List<Number> nums=ints; //compiler error, the type is different.  
    b. Second fragment:



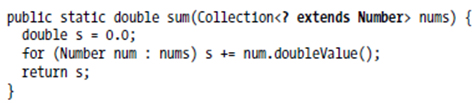
nums.add(3.14);//compiler error, 3.14 is not a type of num



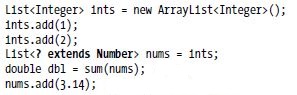
1. Draw a class diagram showing the inheritance relationships among the following types:  
     
   List<Integer>, List<Number>, List<? extends Integer>,   
   List<? extends Number>, List<? super Integer>, List<? super Number>, List<?>, List<Object>



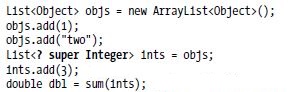
1. Recall the definition of sum given in the slides:



* 1. Is there a compiler error in the following lines of code? If so, where?  
        
     nums.add(3.14);//error



* 1. Is there a compiler error in the following lines of code? If so, where?  
        
     double dbl=sum(1nts);//error



1. Create a generic programming solution to the problem of finding the second smallest element in a list. In other words, devise a public static method secondSmallest so that it can handle the biggest possible range of types.

public static <T extends Comparable> T secondSmallest(List<T> data){  
 if(data.size()<2)  
 return null;  
 T min1=data.get(0);  
 T min2=null;  
 for (int i = 1; i < data.size(); i++) {  
 T e=data.get(i);  
 if(e.compareTo(min1)<0){  
 min2=min1;  
 min1=e;  
 }  
 else if(e.compareTo(min1)>0&&(min2==null||e.compareTo(min2)<0)){  
 min2=e;  
 }  
 }  
 return min2;  
}