CS 61B Spring 2019

Inheritance and Testing

Exam Prep 4: February 11, 2019

1 Playing with Puppers

Suppose we have the Dog and Corgi classes which are a defined below with a few methods but no implementation shown. (modified from Spring '16, MT1)

```
1  public class Dog {
2      public Dog(){ /* D1 */ }
3      public void bark(Dog d) { /* Method A */ }
4  }
5
6  public class Corgi extends Dog {
7      public Corgi(){ /* C1 */ }
8      public void bark(Corgi c) { /* Method B */ }
9      @Override
10     public void bark(Dog d) { /* Method C */ }
11     public void play(Dog d) { /* Method D */ }
12     public void play(Corgi c) { /* Method E */ }
13 }
```

自动 call super(), 等效 call Dog 的 constructor

overloaded

overloaded

	Static Type	Dynamic Type
d	Dog	Corgi
c	Corgi	Corgi
d2	Dog	Dog

For the following main method, at each call to play or bark, tell us what happens at **runtime** by selecting which method is run or if there is a compiler error or runtime error.

```
1  public static void main(String[] args) {
2      Dog d = new Corgi();           Compile-Error   Runtime-Error  C1  D1
3      Corgi c = new Corgi();         Compile-Error   Runtime-Error  C1  D1
4      Dog d2 = new Dog();            Compile-Error   Runtime-Error  C1  D1
5      Corgi c2 = new Dog();          Compile-Error   Runtime-Error  C1  D1
6      Corgi c3 = (Corgi) new Dog();  Compile-Error   Runtime-Error  C1  D1
7
8      d.play(d);        Compile-Error   Runtime-Error   A  B  C  D  E
9      d.play(c);        Compile-Error   Runtime-Error   A  B  C  D  E
10     c.play(d);        Compile-Error   Runtime-Error   A  B  C  D  E
11     c.play(c);        Compile-Error   Runtime-Error   A  B  C  D  E
12     c.bark(d);        Compile-Error   Runtime-Error   A  B  C  D  E
13     c.bark(c);        Compile-Error   Runtime-Error   A  B  C  D  E
14     d.bark(d);        Compile-Error   Runtime-Error   A  B  C  D  E
15     d.bark((int)c);   Compile-Error   Runtime-Error   A  B  C  D  E
16     c.bark((Corgi)d2);Compile-Error   Runtime-Error   A  B  C  D  E
17 }
```

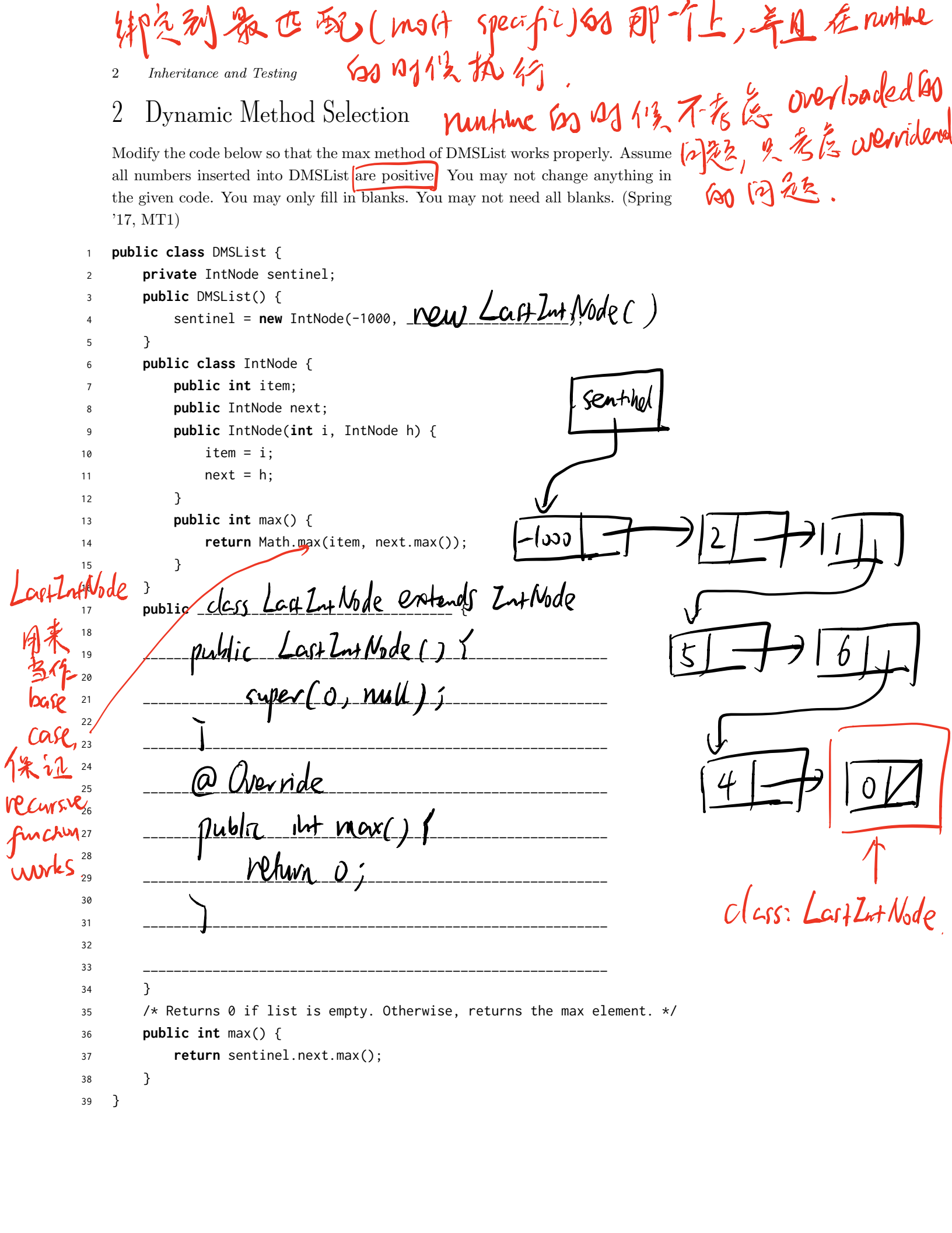
overloaded

★ Overloads are selected at compile time
overrides at runtime.
一旦 method 是 overloaded，则在 compile 的时候就会报

2 Dynamic Method Selection

Modify the code below so that the max method of DMSList works properly. Assume all numbers inserted into DMSList are positive. You may not change anything in the given code. You may only fill in blanks. You may not need all blanks. (Spring '17, MT1)

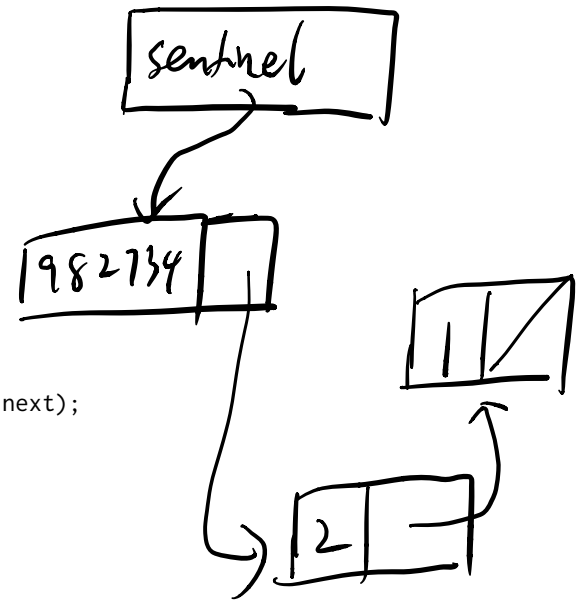
```
1  public class DMSList {
2      private IntNode sentinel;
3      public DMSList() {
4          sentinel = new IntNode(-1000, ____________________);
5      }
6      public class IntNode {
7          public int item;
8          public IntNode next;
9          public IntNode(int i, IntNode h) {
10             item = i;
11             next = h;
12         }
13         public int max() {
14             return Math.max(item, next.max());
15         }
16     }
17     public ____________________ {
18         ____________________
19         ____________________
20         ____________________
21         ____________________
22         ____________________
23         ____________________
24         ____________________
25         ____________________
26         ____________________
27         ____________________
28         ____________________
29         ____________________
30         ____________________
31         ____________________
32
33         ____________________
34     }
35     /* Returns 0 if list is empty. Otherwise, returns the max element. */
36     public int max() {
37         return sentinel.next.max();
38     }
39 }
```



3 SList Debugging and Testing

Consider the SList, a linked list with a sentinel, implementation below. (Spring '16 MT1)

```
1  public class SList {
2    public class IntNode {
3      public int item;
4      public IntNode next;
5      public IntNode(int i, IntNode n) {
6        item = i;
7        next = n;
8      }
9    }
10   private static IntNode sentinel;
11   public SList() {
12     sentinel = new IntNode(982734, null);
13   }
14   public void insertFront(int x) {
15     sentinel.next = new IntNode(x, sentinel.next);
16   }
17   public int getFront() {
18     if (sentinel.next == null) {
19       return -1;
20     }
21     return sentinel.next.item;
22   }
23 }
```



Write a JUnit test that fails on the code above, but would pass on a correct implementation. You may use any JUnit methods like assertEquals, assertNotEquals, assertTrue, assertFalse, etc. Hint: Create at least two instances.

```
1  @Test
2  public void test() {
3    SList s1 = new SList();
4    SList s2 = new SList();
5    s1.insertFront(1);
6    s2.insertFront(2);
7    assertNotEquals(s1.getFront(), s2.getFront());
8    assertEquals(1, s1.getFront());
9    ;
10 }
```