Signature _____ **CSE 8B** **Name _____**

Quiz 4

cs8b ____ **Winter 2015** **Student ID _____**

This quiz is to be taken **by yourself** with closed books, closed notes, no calculators.

Given the following definitions:

```
public interface Speakable {
  public abstract String speak();
}
```

```
public class Dog implements Speakable {
  private static final String
                PUPPY_SPEAK = "";

  public Puppy() {
    // ctor initialization here
  }

  public String speak() {
    return PUPPY_SPEAK;
  }

  public String wag() {
    return "wag tail";
  }
}
```

```
public class Kitty implements Speakable {
  private static final String
                KITTY_SPEAK = "Meow";

  public Kitty() {
    // ctor initialization here
  }

  public String speak() {
    return KITTY_SPEAK;
  }

  public String sleep( int time ) {
    return time + " minute cat nap";
  }
}
```

And the following variable definitions:

```
private Puppy puppy;
private Kitty kitty;
private Speakable speakable;
```

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error".

```
puppy = new Puppy();
kitty = new Kitty();

speakable = kitty;

System.out.println( speakable.sleep( 10 ) );          _____

System.out.println( speakable.speak() );              _____

speakable = puppy;

System.out.println( speakable.wag() );                _____

System.out.println( speakable.speak() );              _____

System.out.println( puppy.wag() );                    _____

System.out.println( puppy.speak() );                  _____

System.out.println( kitty.sleep( 5 ) );               _____

System.out.println( kitty.speak() );                  _____
```

The keyword to inherit from an abstract class is _____

The keyword to inherit from an interface is _____

In the statement

```
new FilledOval( 30, 30, 100, 100, canvas );
```

the first 4 arguments represent _____
(**write the letter representing your answer in the blank above.**)

A	Center point of oval and x diameter and y diameter
B	Upper left corner and lower right corner of bounding box
C	Center point of oval and width and height of bounding box
D	Upper left corner and width and height of bounding box
E	Center point of oval and x radius and y radius

What is the output of this recursive method if it is invoked as `ref.mystery( 11 );`? Draw Stack Frames to help you answer this question.

```
int mystery( int a ) {
  int b = a - 3;

  if ( b >= 5 ) {
    System.out.println( a + " " + b );
    a = b - mystery( b + 1 );
  } else {
    System.out.println( "Stop" );
    b = a + 2;
  }

  System.out.println( a + " " + b );
  return a + b;
}
```

Output

In the current programming assignment, the constructors are to perform deep vs. shallow copies. Fill in the blanks to implement the public Circle(Point center, int radius) { … } constructor with deep copy. Remember to use the appropriate accessor/mutator methods.

```
// in Circle class
public void setCenter( Point p ) { this.center = p; }
public void setRadius( int r ) { this.radius = r; }
```

```
public Circle( Point center, int radius ) {
  _____________________________________  // Invoke superclass ctor with name of this shape

  this.setRadius( radius );              // Initialize radius member in this new Circle
                                         // Initialize center point in this new Circle
  _____________________________________________________________________
}
```

Fill in the blanks for the recursive reverse() method you implemented in the previous programming assignment.

```
public void reverse( int[] array, int low, int high ) {
  if ( _____________________________ == null )
    return;

  if ( _____________________________ >= 1 ) {    // perform swap and recursive call
    int tmp = array[low];

    array[low] = _____________________________________;

    _____________________________________ = tmp;

    reverse( array, _____________________ , _____________________________ );
```

```
  }
}
```

```
public class WordPair implements Comparable{
  private String word;
  private int count;

  public WordPair(String s, int c) {
    this.word = new String(s);
    this.count = c;
  }

  // return 0 if objects are equal
  // return > 0 if calling object is greater than parameter
  // return < 1 if calling object is lesser than parameter
  public int compareTo(Object o) {

    WordPair other = _____________________ o;

    if(  _____________ _____ _____________) {

      return _____________ - _________________;
    }
    else  {

      return ___________________________________;
    }
  }
  public static void main (String [] args) {
    WordPair small = new WordPair( "Hi" , 10 );
    WordPair small2 = new WordPair( "Hi", 10 );
    WordPair big = new WordPair( "Hi", 20 );
    WordPair smallSmall = new WordPair( "Yo", 10 );
    System.out.println( small.compareTo( big ) );
    System.out.println( big.compareTo( small ) );
    System.out.println( small.compareTo( smallSmall ) );
    System.out.println( small.compareTo( small2 ) );
  }
}
```

Output of main
-10
10
-17
0