Poly warmup

Name

1. Consider the following Book and AudioBook classes.

```
public class Book
{
  private int numPages;
  private String bookTitle;

  public Book(int pages, String title)
  {
    numPages = pages;
    bookTitle = title;
  }

  public String toString()
  {
    return bookTitle + " " + numPages;
  }

  public int length()
  {
    return numPages;
  }
}
```

```
public class AudioBook extends Book
{
  private int numMinutes;

  public AudioBook(int minutes, int pages, String title)
  {
    super(pages, title);
    numMinutes = minutes;
  }

  public int length()
  {
    return numMinutes;
  }

  public double pagesPerMinute()
  {
    return ((double) super.length()) / numMinutes;
  }
}
```

Consider the following code segment that appears in a class other than Book or AudioBook.

```
Line 1:  Book[] books = new Book[2];
Line 2:  books[0] = new AudioBook(100, 300, "The Jungle");
Line 3:  books[1] = new Book(400, "Captains Courageous");
Line 4:  System.out.println(books[0].pagesPerMinute());
Line 5:  System.out.println(books[0].toString());
Line 6:  System.out.println(books[0].length());
Line 7:  System.out.println(books[1].toString());
```

Which of the following best explains why the code segment will not compile?

Poly warmup

(A) Line 2 will not compile because variables of type Book may not refer to variables of type AudioBook.

(B) Line 4 will not compile because variables of type Book may only call methods in the Book class.

(C) Line 5 will not compile because the AudioBook class does not have a method named toString declared or implemented.

(D) Line 6 will not compile because the statement is ambiguous. The compiler cannot determine which length method should be called.

(E) Line 7 will not compile because the element at index 1 in the array named books may not have been initialized.

Poly warmup

2. Consider the following class definitions.

```
public class Data
{
  private int x;
  public void setX(int n)
  {
    x = n;
  }
  // ... other methods not shown
}
public class EnhancedData extends Data
{
  private int y;
  public void setY(int n)
  {
    y = n:
  }
  // ... other methods not shown
}
```

Assume that the following declaration appears in a client program.

```
EnhancedData item = new EnhancedData();
```

Which of the following statements would be valid?

I. `item.y = 16;`
II. `item.setY(16);`
III. `item.setX(25);`

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

Poly warmup

3. Consider the following class definitions.

```
public class C1
{
   public C1()
   { /* implementation not shown */ }
   public void m1()
   { System.out.print("A"); }
   public void m2()
   { System.out.print("B"); }
}
public class C2 extends C1
{
   public C2()
   { /* implementation not shown */ }
   public void m2()
   { System.out.print("C"); }
}
```

The following code segment appears in a class other than `C1` or `C2`.

```
C1 obj1 = new C2();
obj1.m1();
obj1.m2();
```

The code segment is intended to produce the output `AB`. Which of the following best explains why the code segment does not produce the intended output?

(A) A compile-time error occurs because `obj1` is declared as type `C1` but instantiated as type `C2`.

(B) A runtime error occurs because method `m1` does not appear in `C2`.

(C) Method `m1` is not executed because it does not appear in `C2`.

(D) Method `m2` is executed from the subclass instead of the superclass because `obj1` is instantiated as a `C2` object.

(E) Method `m2` is executed twice (once in the subclass and once in the superclass) because it appears in both classes.

Poly warmup

4. Consider the following two class definitions.

```
public class Bike
{
   private int numOfWheels = 2;
   public int getNumOfWheels()
   {
     return numOfWheels;
   }
}
public class EBike extends Bike
{
   private int numOfWatts;
   public EBike(int watts)
   {
     numOfWatts = watts;
   }
   public int getNumOfWatts()
   {
     return numOfWatts;
   }
}
```

The following code segment occurs in a class other than `Bike` or `EBike`.

```
Bike b = new EBike(250);
System.out.println(b.getNumOfWatts());
System.out.println(b.getNumOfWheels());
```

Which of the following best explains why the code segment does not compile?

(A) The `Bike` superclass does not have a constructor.

(B) There are too many arguments to the `EBike` constructor call in the code segment.

(C) The first line of the subclass constructor is not a call to the superclass constructor.

(D) The `getNumOfWatts` method is not found in the `Bike` class.

(E) The `getNumOfWheels` method is not found in the `EBike` class.

Poly warmup