☆ Sowtions ☆

Multiple Choice Practice Problems – Mostly Inheritance

#1 What is the output from the following code?

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(2);
list.add(1);
list.add(0);
int n = list.size();
for (int i = 0; i < n; i++)
{
  int v = list.get(i);
  if (v > 0)
    list.add(0, v);
}
System.out.println(list);
```

[2, 1, 0]
↓
[2, 2, 1, 0]
↓
[2, 2, 2, 1, 0]
↓
[2, 2, 2, 2, 1, 0]

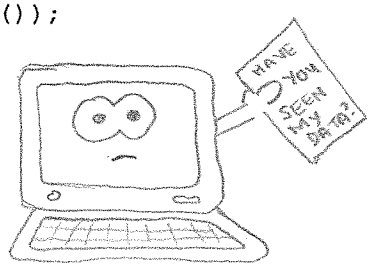
(A) [2, 1, 0, 1, 2]
(B) [2, 1, 0, 2, 1]
(C) [2, 1, 2, 1, 0]
(D) [2, 2, 2, 2, 1, 0]
(E) No output: the program goes into an "infinite" loop and eventually runs out of memory.

Questions #2-3 use the following classes:

```
public class CountDown
{
  private int count;
  public CountDown() { }
  public CountDown(int initCount) { count = initCount; }
  public boolean hasMore() { return count > 0; }
  public int next() { < missing code > }
}

public class CountDown3 extends CountDown
{
  public CountDown3() { < missing code > }
}

public class Test
{
  public static void main(String[] args)
  {
    CountDown c = new CountDown(3);
    while (c.hasMore())
       System.out.print(c.next());
  }
}
```



#4. Consider the following classes:

```
public class A
{
  public String toString() { return "A"; }
}

public class B extends A
{
}

public class C extends B
{
  public String toString() { return super.toString() + "C"; }
}
```

What is the output from

```
A a = new C();
System.out.println(a);
```

variable type ↘ ← object type

(A) A - object type is C so toString() from C class is used
(B) C - super.toString() will add an 'A'
(C) AC
(D) NoSuchMethodException → A has access to a toString() method so this won't happen
(E) B@*dddddd*C, where *dddddd* is a sequence of hex digits

#5 What values are stored in the array `arr` after the following code is executed?

```
int[] arr = {1, 2, 3, 4, 5};
int s = 0;
for (int a : arr)
{
  s += a;
  a = s;
}
```

contents of arr are accessed but can't be modified

(A) 0, 0, 0, 0, 0
(B) 1, 2, 3, 4, 5
(C) 0, 1, 3, 6, 10
(D) 1, 3, 6, 10, 15
(E) 1, 4, 10, 20, 35

#7-9

Questions : use the following classes:

```
public abstract class Movie
{
  private String title;

  public Movie(String t) { title = t; }
  public String getTitle() { return title; }
  public void setTitle(String t) { title = t; }
  public abstract String getRating();
}

public class MovieWithRating extends Movie
{
  private String rating;

  public MovieWithRating(String t, String r)
  {
    super(t);       // Line 1
    rating = r;
  }

  public String getRating() { return rating; }
}
```

#7 Which of the following can replace Line 1 in the `MovieWithRating` constructor, so that

```
MovieWithRating m = new MovieWithRating("Monster House", "PG");
System.out.println(m.getTitle());
```

displays

```
Monster House
```

(A) `title = t;` title is private data in Movie class
(B) `setTitle(t);` MovieWithRating constructor must start w/
(C) `super.setTitle(t);` ← super( ) constructor
(D) `super(); setTitle(t);` will call default movie constructor, the setsTitle (t
(E) None of the above will work

#9 Consider, in addition, the class

```
public class MovieTheater
{
  public static void show(Movie m)
  {
    System.out.println("Showing: " + m.getTitle());
  }

  public static void show(MovieWithRating m)
  {
    System.out.println("Showing: " + m.getTitle() + ", " +
                                                 m.getRating());
  }
}
```

the show() method that gets called depends on the variable type

and the class

```
public class MovieTest
{
  public static void main(String[] args)
  {
    Movie m1 = new MovieWithRating("Monster House", "PG");
    MovieWithRating m2 =
                  new MovieWithRating("Class Action", "R");
    MovieTheater.show(m1);
    MovieTheater.show(m2);
  }
}
```

variable types

What is the result when these classes are compiled, and `main` in `MovieTest` is executed?

(A) Syntax error in the `MovieTheater` class

(B) Syntax error in the `MovieTest` class

(C) Compiles fine and displays
```
Showing: Monster House
Showing: Class Action
```

(D) Compiles fine and displays
```
Showing: Monster House
Showing: Class Action, R
```

only shows title b/c "Monster House" is a Movie variable, so the 1st show method defined is used

(E) Compiles fine and displays
```
Showing: Monster House, PG
Showing: Class Action, R
```