Unit2_final_review

1. Consider the following class.

```
public class SomeMethods

{
public void one(int first)

{ / * implementation not shown * / }

public void one(int first, int second)

{ / * implementation not shown * / }

public void one(int first, String second)

{ / * implementation not shown * / }

}
```

Which of the following methods can be added to the SomeMethods class without causing a compile-time error?

I. public void one(int value)
{ / * implementation not shown * / }

II. public void one (String first, int second)

{ / * implementation not shown * / }

III. public void one (int first, int second, int third)

{ / * implementation not shown * / }

(A) I only
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

Unit2_final_review

2. A pair of number cubes is used in a game of chance. Each number cube has six sides, numbered from 1 to 6, inclusive, and there is an equal probability for each of the numbers to appear on the top side (indicating the cube's value) when the number cube is rolled. The following incomplete statement appears in a program that computes the sum of the values produced by rolling two number cubes.

int sum = / * *missing code* * / ;

Which of the following replacements for /* *missing code* */ would best simulate the value produced as a result of rolling two number cubes?

(A) 2 * (int) (Math.random() * 6)

(B) 2 * (int) (Math.random() * 7)

(C) (int) (Math.random() * 6) + (int) (Math.random() * 6)

(D) (int) (Math.random() * 13)

(E) 2 + (int) (Math.random() * 6) + (int) (Math.random() * 6)

3. The following statement assigns an integer value to `x`.

```
int x = (int)(Math.random() * 5) + 10;
```

Consider the statement that would result if the positions of `5` and `10` were swapped in the preceding statement and the resulting integer were assigned to `y`.

```
int y = (int)(Math.random() * 10) + 5;
```

Which of the following are true statements about how the possible values assigned to `y` differ from the possible values assigned to `x` ?

I. There are more possible values of `x` than there are possible values of `y`.
II. There are more possible values of `y` than there are possible values of `x`.
III. The value assigned to `x` can sometimes be the same as the value assigned to `y`.

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III

Unit2_final_review

4. Consider the following class declaration.

```
public class Student
{
  private String myName;
  private int myAge;

  public Student()
  {  /* implementation not shown */  }

  public Student(String name, int age)
  {  /* implementation not shown */  }

  // No other constructors
}
```

Which of the following declarations will compile without error?

I. Student a = new Student();

II. Student b = new Student("Juan", 15);

III. Student c = new Student("Juan", "15");

(A) I only
(B) II only
(C) I and II only
(D) I and III only
(E) I, II, and III

Unit2_final_review

5. Consider the following class declaration.

```
public class Person
{
  private String myName;
  private int myYearOfBirth;

  public Person(String name, int yearOfBirth)
  {
    myName = name;
    myYearOfBirth = yearOfBirth;
  }

  public String getName()
  {  return myName;  }

  public void setName(String name)
  {  myName = name;  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

Assume that the following declaration has been made.
Person student = new Person ("Thomas", 1995);
Which of the following statements is the most appropriate for changing the name of student from "Thomas" to "Tom" ?

(A) student = new Person ("Tom", 1995);
(B) student.myName = "Tom";
(C) student.getName ("Tom");
(D) student.setName ("Tom");
(E) Person.setName ("Tom");

Unit2_final_review

The following questions refer to the code from the GridWorld case study. A copy of the code is provided below.

Appendix B — Testable API

info.gridworld.grid.Location class (implements Comparable)

public Location(int r, int c)

constructs a location with given row and column coordinates

public int getRow()

returns the row of this location

public int getCol()

returns the column of this location

public Location getAdjacentLocation(int direction)

returns the adjacent location in the direction that is closest to direction

public int getDirectionToward(Location target)

returns the closest compass direction from this location toward target

public boolean equals(Object other)

returns true if other is a Location with the same row and column as this location; false otherwise

Unit2_final_review

public int hashCode()

returns a hash code for this location

public int compareTo(Object other)

returns a negative integer if this location is less than other, zero if the two locations are equal, or a positive integer if this location is greater than other. Locations are ordered in row-major order.

Precondition: other is a Location object.

public String toString()

returns a string with the row and column of this location, in the format (row, col)

Compass directions:

public static final int NORTH = 0;

public static final int EAST = 90;

public static final int SOUTH = 180;

public static final int WEST = 270;

public static final int NORTHEAST = 45;

public static final int SOUTHEAST = 135;

public static final int SOUTHWEST = 225;

public static final int NORTHWEST = 315;

Turn angles:

public static final int LEFT = -90;

public static final int RIGHT = 90;

Unit2_final_review

public static final int HALF_LEFT = -45;

public static final int HALF_RIGHT = 45;

public static final int FULL_CIRCLE = 360;

public static final int HALF_CIRCLE = 180;

public static final int AHEAD = 0;

info.gridworld.grid.Grid<E> interface

int getNumRows()

returns the number of rows, or -1 if this grid is unbounded

int getNumCols()

returns the number of columns, or -1 if this grid is unbounded

boolean isValid(Location loc)

returns true if loc is valid in this grid, false otherwise

Precondition: loc is not null

E put(Location loc, E obj)

puts obj at location loc in this grid and returns the object previously at that location (or null if the location was previously unoccupied).

Precondition: (1) loc is valid in this grid (2) obj is not null

E remove(Location loc)

removes the object at location loc from this grid and returns the object that was removed (or null if the location is unoccupied)

Precondition: loc is valid in this grid

E get(Location loc)

returns the object at location loc (or null if the location is unoccupied)

Precondition: loc is valid in this grid

ArrayList<Location> getOccupiedLocations()

returns an array list of all occupied locations in this grid

ArrayList<Location> getValidAdjacentLocations(Location loc)

returns an array list of the valid locations adjacent to loc in this grid

Precondition: loc is valid in this grid

ArrayList<Location> getEmptyAdjacentLocations(Location loc)

returns an array list of the valid empty locations adjacent to loc in this grid

Precondition: loc is valid in this grid

ArrayList<Location> getOccupiedAdjacentLocations(Location loc)

returns an array list of the valid occupied locations adjacent to loc in this grid

Precondition: loc is valid in this grid

ArrayList<E> getNeighbors(Location loc)

returns an array list of the objects in the occupied locations adjacent to loc in this grid

Precondition: loc is valid in this grid

info.gridworld.actor.Actor class

public Actor()

constructs a blue actor that is facing north

public Color getColor()

returns the color of this actor

public void setColor(Color newColor)

sets the color of this actor to newColor

public int getDirection()

returns the direction of this actor, an angle between 0 and 359 degrees

public void setDirection(int newDirection)

sets the direction of this actor to the angle between 0 and 359 degrees that is equivalent to newDirection

public Grid<Actor> getGrid()

returns the grid of this actor, or null if this actor is not contained in a grid

public Location getLocation()

returns the location of this actor, or null if this actor is not contained in a grid

public void putSelfInGrid(Grid<Actor> gr, Location loc)

puts this actor into location loc of grid gr. If there is another actor at loc, it is removed.

Precondition: (1) This actor is not contained in a grid (2) loc is valid in gr

public void removeSelfFromGrid()

removes this actor from its grid.

Precondition: this actor is contained in a grid

public void moveTo(Location newLocation)

moves this actor to newLocation. If there is another actor at newLocation, it is removed.

Precondition: (1) This actor is contained in a grid (2) newLocation is valid in the grid of this actor

public void act()

reverses the direction of this actor. Override this method in subclasses of Actor to define types of actors with different behavior

public String toString()

returns a string with the location, direction, and color of this actor

info.gridworld.actor.Rock class (extends Actor)

public Rock()

constructs a black rock

public Rock(Color rockColor)

constructs a rock with color rockColor

public void act()

overrides the act method in the Actor class to do nothing

info.gridworld.actor.Flower class (extends Actor)

public Flower()

constructs a pink flower

public Flower(Color initialColor)

constructs a flower with color initialColor

public void act()

causes the color of this flower to darken

6. Consider the following method that is intended to move the parameter anActor to a different grid that is referred to by the parameter newGrid. The location of anActor in newGrid should be the same as the location that anActor had occupied in its original grid.

```
/** Moves anActor to newGrid in the same location it occupied in its original grid.
* @param anActor the actor to be moved
* @param newGrid the grid in which the actor is to be placed
*/
public void moveActorToNewGrid(Actor anActor, Grid<Actor> newGrid)
{
Grid<Actor> oldGrid = anActor.getGrid();
Location loc = anActor.getLocation();
/* missing code */
}
```

Which of the following can be used to replace /* *missing code* */ so that moveActorToNewGrid will work as intended?

(A)
```
anActor.putSelfInGrid(newGrid, loc);
anActor.removeSelfFromGrid();
```

(B)
```
oldGrid.remove(loc);
anActor.putSelfInGrid(newGrid, loc);
```

(C)
```
anActor.removeSelfFromGrid();
anActor.putSelfInGrid(newGrid, loc);
```

(D)
```
oldGrid.remove(loc);
newGrid.put(anActor, loc);
```

(E)
```
newGrid.put(anActor, loc);
oldGrid.remove(loc);
```

---

7. Consider the following method.

```
public static String scramble(String word, int howFar)
{
  return word.substring(howFar + 1, word.length()) +
         word.substring(0, howFar);
}
```

What value is returned as a result of the call scramble("compiler", 3)?

(A) "compiler"

(B) "pilercom"

(C) "ilercom"

(D) "ilercomp"

(E) No value is returned because an IndexOutOfBoundsException will be thrown.

8. Consider the following method, which is intended to return true if at least one of the three strings s1, s2, or s3 contains the substring "art". Otherwise, the method should return false.

```
public static boolean containsArt(String s1, String s2, String s3)
{

  String all = s1 + s2 + s3;

  return (all.indexOf("art") != -1);

}
```

Which of the following method calls demonstrates that the method does not work as intended?

Unit2_final_review

(A) containsArt ("rattrap", "similar", "today")

(B) containsArt ("start", "article", "Bart")

(C) containsArt ("harm", "chortle", "crowbar")

(D) containsArt ("matriculate", "carat", "arbitrary")

(E) containsArt ("darkroom", "cartoon", "articulate")

9. Consider the following class definition.

```
public class Bird
{
     private String species;
     private String color;
     private boolean canFly;
     public Bird(String str, String col, boolean cf)
     {
         species = str;
         color = col;
         canFly = cf;
     }
}
```

Which of the following constructors, if added to the `Bird` class, will cause a compilation error?

Unit2_final_review

(A)
```
public Bird()
{
     species = "unknown";
     color = "unknown";
     canFly = false;
}
```

(B)
```
public Bird(boolean cf)
{
     species = "unknown";
     color = "unknown";
     canFly = cf;
}
```

(C)
```
public Bird(String col, String str)
{
     species = str;
     color = col;
     canFly = false;
}
```

(D)
```
public Bird(boolean cf, String str, String col)
{
     species = str;
     color = col;
     canFly = cf;
}
```

(E)
```
public Bird(String col, String str, boolean cf)
{
     species = str;
     color = col;
     canFly = cf;
}
```

Unit2_final_review

10. Consider the following attempts at method overloading.

I.

```
public class Overload
{
     public int average(int x, int y)
     { /* implementation not shown */ }

     public int average(int value1, int value2)
     { /* implementation not shown */ }

     // There may be instance variables, constructors,
     // and methods that are not shown.
}
```

II.

```
public class Overload
{
     public int average(int x, int y)
     { /* implementation not shown */ }

     public int average(int x, int y, int z)
     { /* implementation not shown */ }

     // There may be instance variables, constructors
     // and methods that are not shown.
}
```

III.

```
public class Overload
{
     public int average(int x, int y)
     { /* implementation not shown */ }

     public int average(double x, double y)
     { /* implementation not shown */ }

     // There may be instance variables, constructors,
     // and methods that are not shown.
}
```

Which of the attempts at method overloading will compile without error?

Unit2_final_review

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

11. Consider the `processWords` method. Assume that each of its two parameters is a `String` of length two or more.

```
public void processWords(String word1, String word2)
{
     String str1 = word1.substring(0, 2);
     String str2 = word2.substring(word2.length() - 1);
     String result = str2 + str1;
     System.out.println(result.indexOf(str2));
}
```

Which of the following best describes the value printed when `processWords` is called?

(A) The value `0` is always printed.
(B) The value `1` is always printed.
(C) The value `result.length() - 1` is printed.
(D) A substring containing the last character of `word2` is printed.
(E) A substring containing the last two characters of `word2` is printed.

12. Which of the following statements assigns a random integer between 25 and 60, inclusive, to `rn` ?

(A) `int rn = (int) (Math.random() * 25) + 36;`
(B) `int rn = (int) (Math.random() * 25) + 60;`
(C) `int rn = (int) (Math.random() * 26) + 60;`
(D) `int rn = (int) (Math.random() * 36) + 25;`
(E) `int rn = (int) (Math.random() * 60) + 25;`

13. Consider the following code segment. Assume that `a` is greater than zero.

```
int a = /* value not shown */;
int b = a + (int) (Math.random() * a);
```

Which of the following best describes the value assigned to `b` when the code segment is executed?

(A) `a`
(B) `2 * a`
(C) A random integer between `0` and `a - 1`, inclusive
(D) A random integer between `a` and `2 * a`, inclusive
(E) A random integer between `a` and `2 * a - 1`, inclusive

14. Assume that the following variable declarations have been made.

```
double d = Math.random();

double r;
```

Which of the following assigns a value to r from the uniform distribution over the range $0.5 \leq r < 5.5$?

(A) r = d + 0.5;

(B) r = d + 0.5 * 5.0;

(C) r = d * 5.0;

(D) r = d * 5.0 + 0.5;

(E) r = d * 5.5;

Directions: Select the choice that best fits each statement. The following question(s) refer to the following information.

Consider the following partial class declaration.

```
public class SomeClass
{
  private int myA;
  private int myB;
  private int myC;

  // Constructor(s) not shown

  public int getA()
  {  return myA;  }

  public void setB(int value)
  {  myB = value;  }
}
```

15. The following declaration appears in another class.
SomeClass obj = new SomeClass ( );
Which of the following code segments will compile without error?

Unit2_final_review

(A) int x = obj.getA ( );

(B)
```
int x;
obj.getA (x);
```

(C) int x = obj.myA;

(D) int x = SomeClass.getA ( );

(E) int x = getA(obj);