

CSE 11

UCSD

Final

March 22, 2002

CLOSED BOOK

WARNING: These answers have not been checked carefully

Last Name: _____ ANSWERS _____

First and Middle Names: _____

Login Name (USE ALL CAPS): CS11W _____

City of your birth _____

Signature _____

IMPORTANT: Put your name on every sheet. You may not get credit for sheets that do not have your name.

The entire exam is worth 165 points.

Multiple choice: _____ 90 _____

31: _____ 15 _____

32: _____ 15 _____

33: _____ 15 _____

34: _____ 10 _____

35: _____ 20 _____

TOTAL _____

Multiple Choice

Circle the correct answer. Graded as follows: Correct 3 points; no answer, incorrect answer (or more than one answer) 0 point. No negative points. Answer with the ONE BEST answer. Do not mark more than one answer for a question.

1. The type `boolean`

- >>a. is a primitive type.
- b. is a class type.
- c. is a wrapper type.
- d. all of the above.
- e. none of the above.

2. The `SavitchIn` class

- a. is defined in the library `javax.swing`
- b. is defined in the library `java.awt`.
- c. is part of the core of Java and not in any library
- >>d. is written in Java, but is not part of the official Java language.
- e. none of the above.

3. The `WindowDestroyer` class

- a. is defined in the library `javax.swing`
- b. is defined in the library `java.awt`.
- c. is part of the core of Java and not in any library
- >>d. is written in Java, but is not part of the official Java language.
- e. none of the above.

4. Which of the following are reference types

- a. `char`
- b. `Character`
- c. `char[]`
- d. all of the above.
- e. a and b
- f. a and c.
- >>g. b and c
- h. none of the above.

5. Which of the following are reference types

- a. `double`
- b. `boolean`
- >>c. `String`
- d. all of the above
- e. none of the above.

6. A mutator method is used
- a. to find out the value of an instance variable.
 - >>b. to change the value of an instance variable.
 - c. to initialize instance variables.
 - d. all of the above.
 - e. none of the above.
7. An instance variable of a class
- a. can be of a primitive type.
 - b. can be of a class type.
 - c. can be an array type.
 - >>d. all of the above.
 - e. none of the above.
8. A default constructor
- a. is produced automatically by Java for every class.
 - b. must be defined for every class or the class will not compile.
 - >>c. is a constructor with no parameters.
 - d. all of the above.
 - e. none of the above.
9. You may overload a method name by giving two definition
- a. that return different types.
 - >>b. that have different numbers of parameters.
 - c. for which one is static and one is not.
 - d. all of the above.
 - e. none of the above.
10. Suppose that in some class definition you have the following method invocation:
- ```
doStuff(5, 4.3);
```
- Below are the headings of method definition in the class. Which method definition will be used in this invocation?
- a. `public void doStuff(double x, double y)`
  - >>b. `private void doStuff(int x, double y)`
  - c. `public static void doStuff(int x, int y).`
  - d. none of the above.

11. A static method

- a. cannot be invoked with a calling object.
- >>b. can be invoked using the class name in place of a calling object.
- c. must be a void method.
- d. all of the above
- e. none of the above.

12. A private static instance variable of a class

- a. can be referenced by name by all the objects of the class..
- b. can be used to pass information from one object of the class to another object of the class..
- c. has only one copy for all objects of the class.
- >>d. all of the above
- e. none of the above.

13. The value `null` may be assigned

- a. to any variable.
- b. to any variable of a primitive type.
- >>c. to any variable of a class type.
- d. none of the above.

14. The size of an array

- a. is determined by the array type.
- >>b. is determined when the array is created using the `new` operator..
- c. can change when more indexed variables are needed.
- d. none of the above.

15. Suppose `Species` is a class type and you want to create an array of 10 objects of type `Species`. How many times must you use the operator `new`.

- a. one
- b. nine
- c. ten
- >>d. eleven
- e. none of the above.

16. Which of the following are true of vectors?

- a. A vector can increase its size until the size reaches the capacity of the vector.
- b. A vector can increase its size without limit (other than running out of memory)..
- c. has base type `Object`..
- d. all of the above.
- e. a and b
- f. a and c
- >>g. b and c.
- h. none of the above

17. A method may return a value of type

- a. `int[]`
- b. `String[]`
- c. `Integer[]`
- >>d. all of the above
- e. none of the above.

18. Suppose `Child` is a derived class of the `Parent`. Suppose a method named `doStuff()` is defined in `Parent` and has its definition overridden (changed) in the definition of class `Child`. Suppose `c` is an object of type `Child`, what definition of `doStuff()` is used in `c.doStuff()`:

- >>a. The one given in the definition of `Child`.
- b. The one given in the definition of `Parent`.
- c. It depends on the type of the variable used to name `c`.
- d. You need more information to decide which definition of `doStuff()` is used.
- e. none of the above.

19. Suppose `Child` is a derived class of `Parent` and `Parent` is a derived class of `GrandParent`. Which of the following are true?

- a. An object of the class `Child` is of type `Parent`.
- b. An object of the class `Child` is of type `GrandParent`.
- c. An object of the class `Child` is of type `Child`.
- >>d. all of the above
- e. none of the above

20. Suppose `Child` is a derived class of `Parent` and `doStuff()` is a private method in the class `Parent`. Which of the following are true?

- a. `doStuff()` may be used in the definition of methods in `Child`.
- >>b. `doStuff()` may not be used in the definition of methods in `Child`.
- c. In some, but not all cases, `doStuff()` may be used in the definition of methods in `Child`..
- d. none of the above

21. Every exception class a programmer defines

- a. has a method named `getMessage`.
- b. must be a derived class of some other exception class
- c. is a descendent class of the class `Object`.
- >>d. all of the above.
- e. none of the above

22. Which of the following are methods of the class `Object`?

- a. `toString`.
- b. `equalsIgnoreCase`.
- c. `equals`
- d. all of the above
- e. a and b
- >>f. a and c.
- g. b and c
- h. none of the above

23. When reading from a file (with one of the classes discussed in this course) how can your program detect when it has reached the end of a the file?

- a. If the file is a text file, then an `EOFException` is thrown when your code attempts to read beyond the end of the file.
- >>b. If the file is a binary file, then an `EOFException` is thrown when your code attempts to read beyond the end of the file.
- c. both a. and b.
- d. none of the above

24. Which of the following are methods in the class `File`?

- a. `canRead`
- b. `canWrite`
- c. `writeUTF`.
- d. all of the above.
- >>e. a and b
- f. a and c
- g. b and c.
- h. none of the above

25. If a number is written to a file using the method `writeInt` of the class `DataOutputStream`, then it should be read with which of the following methods?

- >>a. `readInt`
- b. `readDouble`
- c. Either `readInt` or `readDouble`.
- d. `readUTF`
- e. none of the above

26. Which of the following are true of interfaces?

- a. An interface is a class.
- >>b. An interface is a type.
- c. An interface is an object.
- d. all of the above.
- e. a and b
- f. a and c
- g. none of the above

27. The method `setActionCommand` is in which of the following?

- a. The class `JButton`.
- b. The class `JMenuItem`.
- c. The class `AbstractButton`.
- >>d. all of the above.
- e. none of the above

28. Which of the following can be added to a `JMenu`?

- a. A `JMenuItem`.
- b. A `JMenu`.
- >>c. both a and b.
- d. none of the above.

29. Which of the following are layout managers?

- a. `BoxLayout`.
- b. `BorderLayout`.
- c. `CardLayout`.
- >>d. all of the above.
- e. none of the above.

30. The content pane of a `JFrame` is of type

- >>a. `Container`.
- b. `JComponent`.
- c. `JPanel`.
- d. all of the above.
- e. none of the above.

## Short Answer

\*In many cases the amount of space allowed for answer is much larger than you need. Do not take that as a guide to how long the answer is.

\*Style will count

\*The following class definition is used in some of the following questions. It is taken directly from the book.

```

/*****
 *Class for data on endangered species.
 *****/
public class Species
{
 private String name;
 private int population;
 private double growthRate;

 public Species()
 {
 name = null;
 population = 0;
 growthRate = 0.0;
 }

 public Species(String theName, int thePopulation, double theGrowthRate)
 {
 set(theName, thePopulation, theGrowthRate);
 }

 public void readInput()
 {
 System.out.println("What is the species' name?");
 name = SavitchIn.readLine();
 System.out.println(
 "What is the population of the species?");
 population = SavitchIn.readLineInt();
 while (population < 0)
 {
 System.out.println("Population cannot be negative.");
 System.out.println("Reenter population:");
 population = SavitchIn.readLineInt();
 }
 System.out.println(
 "Enter growth rate (percent increase per year):");
 growthRate = SavitchIn.readLineDouble();
 }

 public void writeOutput()
 {
 System.out.println("Name = " + name);
 System.out.println("Population = " + population);
 System.out.println("Growth rate = " + growthRate + "%");
 }

/*****
 *Precondition: years is a nonnegative number.
 *Returns the projected population of the calling object after the specified number of years.
 *****/
 public int projectedPopulation(int years)

```

```

{
 double populationAmount = population;
 int count = years;

 while ((count > 0) && (populationAmount > 0))
 {
 populationAmount = (populationAmount +
 (growthRate/100) * populationAmount);
 count--;
 }
 if (populationAmount > 0)
 return (int)populationAmount;
 else
 return 0;
}

public void set(String newName, int newPopulation, double newGrowthRate)
{
 name = newName;
 if (newPopulation >= 0)
 population = newPopulation;
 else
 {
 System.out.println("ERROR: using a negative population.");
 System.exit(0);
 }
 growthRate = newGrowthRate;
}

public String getName()
{
 return name;
}

public int getPopulation()
{
 return population;
}

public double getGrowthRate()
{
 return growthRate;
}

public boolean equals(Species otherObject)
{
 return ((name.equalsIgnoreCase(otherObject.name))
 && (population == otherObject.population)
 && (growthRate == otherObject.growthRate));
}
}

```

31. Give the definition of a **STATIC** method named `doublePop` that can be added to the class `Species`. The method `doublePop` has one parameter called `model` of type `Species` and returns an `int` value that is twice as large as the population of the object `model`. For full credit, you must do it without using any accessor methods. You will get partial credit for doing it with one or more accessor methods. (Hint: It is very short.)

ANSWER (15 points)

```
public static int doublePop(Species model)
{
 return 2*model.population;
}
```

the following is worth 10 points

```
public static int doublePop(Species model)
{
 return 2*model.getPopulation();
}
```

32. Write some Java code that will create a vector of `Species` of size 25 and will read input values for each of the 25 species from the keyboard and store them in the vector elements. (The class `Species` is given in these exam pages.) Do not write a complete program, but do declare all variables and use the `new` operator to create all objects. Use the method `readInput` to read the data. You do NOT need to give any import statement(s).

ANSWER (15 points)

```
import java.util.*; //not needed for answer

Vector v = new Vector(25); //25 can be omitted
int i;
for (i = 0; i < 25; i++)
 v.addElement(new Species());
System.out.println("Enter input for 25 species:");
for (i = 0; i < 25; i++)
 ((Species)v.elementAt(i)).readInput();
```

33. Give the definition of a class called `MatingSpecies` which is a derived class of the class `Species` (given in these pages). The class `MatingSpecies` has one added instance variable named `femalePopulation` which is of type `int`. The class `MatingSpecies` has one constructor that has THREE parameters, one for the name of the species, one for the total population of the species, and one for the growth rate of the species. The three inherited instance variables are set to the three parameter values. The new instance variable `femalePopulation` is set equal to one half of the total population. If the total population is an odd number, such as 5, then the instance variable `femalePopulation` is set to a value equal to one half of the total population rounding up to a whole number, such as 3 for a total population of 5. The class `MatingSpecies` has no other constructors and no other methods of any kind other than those inherited from the class `Species`.

ANSWER (15 points)

```
public class MatingSpecies extends Species
{
 private int femalePopulation;

 public MatingSpecies(String theName, int thePopulation, double theGrowthRate)
 {
 super(theName, thePopulation, theGrowthRate);
 femalePopulation = thePopulation/2 + thePopulation%2;
 }
}
```

Alternatives

can use  
`set(theName, thePopulation, theGrowthRate);`  
 instead of  
`super(theName, thePopulation, theGrowthRate);`

can use  
`getPopulation()`  
 in place of  
`thePopulation` (the parameter)

CANNOT use  
`population/2`

34. Define a derived class of the class `JPanel` that is called `PinkJPanel`. An object of the class `PinkJPanel` can be used just like we used objects of the class `JPanel`, but an object of the class `PinkJPanel` is pink in color (unless you explicitly change its color). The class `PinkJPanel` will only have one constructor, namely the default constructor. You do not need to give the import statements. (Hint: This is very easy.)

ANSWER (10 points)

```
import javax.swing.*; //not needed for answer
import java.awt.*; //not needed for answer

public class PinkJPanel extends JPanel
{
 public PinkJPanel()
 {
 setBackground(Color.pink);
 }
}
```

35. Write a complete Java program, which when run will display a Swing GUI that has a display like the following

```
1 2 3
4 5 6
```

The display is inside the window (that is, inside the `JFrame`). Use the `GridLayout` manager to position the numbers. Make each number a label. When the close window button is clicked, the program ends. You may assume the `WindowDestroyer` class, as defined in the text, is in the directory where the program is run (so, you can use but do not need to define the class `WindowDestroyer`). Your window does not do or show anything else. DO NOT embellish your window with anything like a title, colors, or anything else that is not asked for. You do NOT need to give the import statements. For full credit you must use a nice clean for-loop that adds the 6 number labels using the integers 1 through 6. Partial credit will be given for less elegant solutions that not use a for-loop or do not use it in a nice clean way.

ANSWER (20 points)

```
public class ExamWindow extends JFrame
{
 public ExamWindow()
 {
 Container content = getContentPane();
 content.setLayout(new GridLayout(2, 3));
 for (int i = 1; i <= 6; i++)
 content.add(new JLabel(Integer.toString(i)));
 addWindowListener(new WindowDestroyer());
 }

 public static void main(String[] args)
 {
 ExamWindow gui = new ExamWindow();
 gui.setVisible(true);
 }
}
```

Six explicit adds, something like,

```
content.add(new JLabel("1"));
```

or

```
JLabel aLabel = new JLabel("1");
```

```
content.add(aLabel);
```

take off 5 points, more if it is not correct.