**CS390 Fundamental Programming Practices**

**Midterm Exam**

**Student Name** \_\_\_\_\_\_\_\_\_\_\_\_ **Student ID** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part1 (score 10)** | **Part2 (score 5)** | **Part3 (score 16)** | **Part4 (score 9)** | **Total 40** |
|  |  |  |  |  |

**PART1 – True/False, place your answer in front of the question number**

1. An abstract class or interface can have no abstract method.
2. We can override static method. Static variables cannot be accessed by instance methods.
3. Overloaded methods CANNOT be in the same class.
4. Overloading is determined during the runtime time. Overriding determined during the compile time.
5. A member inner class has access to all fields and methods in its enclosing class.
6. If a class is declared abstract, it cannot be instantiated.
7. An overriding method access modifier cannot be default if the overridden method in its superclass access modifier is public
8. The subclass of an abstract class should always override the abstract methods.
9. Java class can implement more than one interface for the aspects of multiple inheritance.
10. A subclass constructor must call at least one constructor in superclass implicitly or explicitly.

**PART2 - Blank Space Filling**

Please, fill out the blank spaces.

1. We use \_\_\_\_\_\_\_\_\_\_\_\_ keyword to indicate that you are accessing the superclass’s version of certain method.
2. \_\_\_\_\_\_\_\_\_\_\_\_ keyword can be used to prevent class to be inherited. \_
3. refers to the ability of an object to take on many forms.
4. Fill the blank with the correct keyword.

\_\_\_\_\_\_\_\_\_\_\_\_ class Plant {

public abstract String getType();

}

1. You must clone all the objects referenced by all reference variables of an object to get a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ cloning.

**PART3 - Multiple Choice, place your answer in front of the question number**

1. Which one is NOT a correct way to override the following method of Parent class.

public class Parent {

void display(){

System.out.println("Parent class display method.");

}

}

1. @Override

private void display() {

System.out.println("Child class display method");

}

1. @Override

public void display() {

System.out.println("Child class display method");

}

1. @Override

protected void display() {

System.out.println("Child class display method");

}

d. @Override

void display() {

System.out.println("Child class display method");

}

1. In which case a class should NOT inherit from another class.
2. Two classes can have IS-A relationship.
3. A subclass instance can be used whenever a superclass instance is expected.
4. The two classes ONLY have some methods in common.
5. One of the class is a special type of the other class, and the other class can be derived to more types.
6. Select one correct answer to fix this code.

public abstract class Vehicle {

public final abstract String getType();

}

1. Remove abstract keyword from class declaration.
2. Remove abstract keyword from getType() method declaration.
3. Add method body to getType() method.
4. Remove final keyword from getType() method declaration.
5. Which one can be overridden in a sub class?
6. public void printInfo() {

System.out.println("Super Class version of printInfo method.");

}

1. private void printInfo() {

System.out.println("Super Class version of printInfo method.");

}

1. public final void printInfo() {

System.out.println("Super Class version of printInfo method.");

}

1. public static void printInfo() {

System.out.println("Super Class version of printInfo method.");

}

1. Which one is NOT from Object class.
2. compareTo ()
3. hashCode()
4. clone()
5. equals()
6. What is the output of the following code.

interface Plant {

public default void grow() {

System.out.println("Plant is growing.");

}

}

class Flower {

public void grow() {

System.out.println("Flower is growing");

}

}

public class SunFlower extends Flower implements Plant{

public static void main(String[] args) {

SunFlower sunflower = new SunFlower();

sunflower.grow();

}

}

1. Plant is growing
2. Flower is growing
3. Nothing will be printed.
4. Compiler Error
5. What is the output of below program?

class Bar{

Bar(){ }

Bar(int i){

this(5, 15);

System.out.println(i);

}

Bar(int x, int y){

System.out.println(x \* y);

}

}

public class Foo extends Bar{

Foo(){

super(5, 10);

}

Foo(int x){

super(5);

System.out.println(x);

}

public static void main(String args[]){

new Foo(5);

}

}

a. 50 b. 5, 75, 5 c. 5, 5, 75 d. 75, 5, 5

1. Which one CAN NOT to print out Child run for 1 mile and rest for 1 hour?

class Parent{

void run() {

System.out.println("Run for 5 miles.");

}

}

public class Child extends Parent{

@Override

void run() {

System.out.println("Run for 1 mile.");

}

void rest() {

System.out.println("Rest for 1 hour.");

}

}

a. Child c = new Child();

Parent p = c;

p.run();

c.rest();

b. Parent p = new Parent();

p.run();

((Child)p).rest();

c. Child c = new Child();

c.run();

c.rest();

d. Parent c = new Child();

c.run();

((Child)c).rest();

**PART4 - Code implementation**

1. Write an implementation of the classes Cent, Dime and Quarter.
2. Write an output of the Main class.

public interface Money {

public int getAmount();

}

public abstract class Coin {

public abstract String getColor();

}

public class Main {

public static void main(String[] args) {

Coin[] coins = {new Cent(), new Dime(), new Quarter(), new Quarter()};

for(Coin coin : coins) {

System.*out*.println(coin.getClass().getSimpleName() + " is " + coin.getColor());

}

Money[] money = {new Cent(), new Dime(), new Quarter(), new Quarter()};

int sum = 0;

for(Money m: money) {

sum += m.getAmount();

}

System.*out*.println("Sum amount is " + sum);

}

}

public class Dime extends Coin implements Money

{

//IMPLEMENT

}

public class Quarter extends Coin implements Money

{

//IMPLEMENT

}

public class Cent extends Coin implements Money

{

//IMPLEMENT

}

1. Implement two methods below to make the main method print out “true”

public class Singleton {

private static class MySingletonHolder {

//IMPLEMENT

}

public static Singleton getInstance() {

//IMPLEMENT

}

public static void main(String[] args) {

Singleton s1 = Singleton.getInstance();

Singleton s2 = Singleton.getInstance();

System.out.println(s1 == s2);

}

}

4.Write a recursive solution to print out each letter in the string “abcde” in reverse order