Practice Questions Midterm

```
class Alpha {
	public void foo(){
		System.out.println("Alpha");
	}
}
public class Beta extends Alpha {
	protected void foo(){
		System.out.println("Beta");
	}
	//DO NOT ALTER MAIN
	public static void main(String[] args){
		Alpha a;
		a = new Beta();
		a.foo();

	}
}
public class C {
	public static void main(String[] args){
		Alpha a = new Beta();
		a.foo();
	}
}
```

Error Msg: Beta.java:7:

```
public class Egg {
	private int i;
	Yolk y;
	class Yolk {
		public  void setI(int value){ i = value;}
	}
	public static void main(String[] args){
		Egg e = new Egg();
		e.y.setI(5);
		System.out.println("i="+e.i);
	}
}
```

Desired Output: **i=5**

Error Msg: (Runtime) Egg.java:9

```
class Father {
	public final void talk(){System.out.println("Manners");}
}
public class Son extends Father {
	public final void talk(){
		System.out.println("Sloppy");
	}
	public static void main(String[] args){
		Son s = new Son();
		s.talk();
	}
}
```

Desired Output:
Sloppy
Manners

Error Msg: Son.java: 5

```
class Animal {
	public  void walk(){
		System.out.println("Walk like an animal");
	}
}
class Tiger extends Animal {
	public void walk(){
		System.out.println("Walk like a Tiger");
	}
}
public class Cat extends Tiger {
	public void walk(){
		System.out.println("Walk like a Cat");
	}
	public static void main(String[] args){
		Animal c = new Cat();
		c.walk();
	}
}
```

Desired Output:
Walk like an animal
Walk like a Tiger
Walk like a Cat

Error Msg: none. Output NOT CORRECT

```
interface Face {
	void eyes();
	void mouth();
	void nose();
	void ears();
}

public class Head implements Face {
	public void eyes(){}
	public void mouth(){}
	public void nose(){}
}
```

Error Msg: Head.java:8

package Friendly;

```
public class Me{
	void greet(){ System.out.println("Hello");}
}
```

---

package Protected;

```
import Friendly.Me;
public class Friend extends Me {

	protected void talk(){
		greet();
		System.out.println("Hi");
	}
	public static void main(String[] args){
		Friend f = new Friend();
		f.talk();
	}
}
```

Desired Ouput:
Hello
Hi

Error Msg: Protected.Friend.java:5

```
public class Awake {
	public void Awake(String time){
		System.out.println("Time is "+time);
	}
	public static void main(String[] args){
		Awake a = new Awake("up!");
	}
}
```

Desired Output: **Time is up!**

Error Msg: Awake.java:6

```
class Five {
	public void number(int x){
		System.out.println(x+5);
	}
}
 public class Four extends Five {
	public void number(double x){
		System.out.println(x+4);
	}
	public static void main(String[] args){
		Five f = new Four();
		f.number(3.141579);
	}
}
```

Error Msg Four.java: 12

```
interface GeneralI {}

class Scarry implements GeneralI {
	public void boo(){
		System.out.println("Boo");
	}
}

class UnHappy extends Scarry {
	public void boo(){
		System.out.println("Hoo");
	}
}
public class Main {
	public static void main(String[] args){
		foo(new UnHappy());
	}
	private static void foo(GeneralI x){
		x.boo();
	}
}
```

Error Msg:Main.java:19

```
public class RaceCar {
	String driverName;
	public static void main(String[] args){
		if (args[0] != null)
			driverName = args[0];
	}
}
```

Error Msg:RaceCar.java:5

```
interface Phone {
	void number();
	void message();
}
abstract class FancyPhone implements Phone {
	public void number(){
		System.out.println("The number you are calling is not available");
	}
	public void message(String m){
		System.out.println("The message "+m);
	}
}
public class ExecPhone extends FancyPhone implements Phone {
	public static void main(String[] args){
		ExecPhone exec = new ExecPhone();
		exec.number();
		exec.message("Hello");
	}
}
```

Error Msg: ExecPhone.java:13

```
public class Big {
	private Small s = new Small();
	private class Small {
		public int size = 1;
	}
	public static void main(String[] args){
		Big b = new Big();
		b.size;
	}
}
```

Error Msg:Big.java:8

package comp2525;

```
public class A{
	int x;
	protected double y;
}
```

---

package comp1510;

```
import comp2525.A;
import comp2525.B;

public class C extends A{
	public C(int x, int y) {
		this.x = x;
		this.y = y;
	}
}
```

Error Msg:C.java:7

Give the output

```
public class Student {
        int numStudents = 0;
        public Student(){
                numStudents++;
        }
        public static void main(String[] args){
                Student s1, s2, s3;
                s1= new Student();
                s2= new Student();
                s3= new Student();
                System.out.println("Total number of students is "+s3.numStudents);
        }
}
```

<u>Output</u>

```
class Cup {
  Cup(int marker) {System.out.println("Cup(" + marker + ")");}
  void f(int marker) {System.out.println("f(" + marker + ")");}
}
class Cups {
  static Cup c1;
  static Cup c2;
  static {
    c1 = new Cup(1);
    c2 = new Cup(2);
  }
  Cups() {System.out.println("Cups()");}
}
public class ExplicitStatic {
    static Cups x = new Cups();
    static Cups y = new Cups();
  public static void main(String[] args) {
    System.out.println("Inside main()");
  }
}
```

<u>Output</u>

```
class Shape {
	void draw(){
		System.out.println("no shape yet");
	}
	Shape(){
		System.out.println("creating shape");
		draw();
		System.out.println("finished drawing shape");
	}
}
public class Circle extends Shape {
	int radius = 1;
	Circle(int r){
		radius = r;
		System.out.println("Circle has radius = "+radius);
	}
	void draw(){
		System.out.println("Draw Circle, radius = "+radius);
	}
	public static void main(String[] args){
		new Circle(5);
	}
}
```

<u>Output</u>

```
interface Greeting {
	String sendGreeting();
}
class Hello implements Greeting {
	public String sendGreeting(){
		return “Hello”;
	}
}
class Goodbye implements Greeting {
	public String sendGreeting(){
		return “Goodbye”;
	}
}
public class Main {
	static Greeting g1, g2;
	public static void swap(Object a, Object b){
		Object temp = a;
		a = b;
		b = temp;
	}
	public static void main(String[] args){
		g1 = new Hello();
		g2 = new Goodbye();
		System.out.println(g1.sendGreeting());
		swap(g1, g2);
		System.out.println(g1.sendGreeting());
	}
}
```

<u>Output</u>

```
interface Animal {
	void draw();
}
class Dog implements Animal {
	public void draw(){
		System.out.println("Dog");
	}
}
class Kohkoh extends Dog {
	public void draw(){
		super.draw();
		System.out.println("Kohkoh");
	}
}
class Mastif extends Dog implements Animal {
	public void draw(){
		System.out.println("Mastif");
	}
}
class Ridgeback extends Kohkoh {
	Mastif m = new Mastif();
	public void draw(){
		m.draw();
		super.draw();
		System.out.println("Ridgeback");
	}
}
public class Kennel {
	public static void main(String[] args){
		Animal a;
		Dog d;
		d = new Mastif();
		a = new Ridgeback();
		a.draw();
		d.draw();
		a = d;
		a.draw();
	}
}
```

Output

```
public class Alphabet {
	public void display(){ System.out.print("Alpha");}
}
public class A extends Alphabet {
	public void display(){ System.out.print("A");}
}
public class B extends Alphabet {
	public void display(){ System.out.print("B");}
}
public class C extends Alphabet {
	public void display(){ System.out.print("C");}
}
public class D extends Alphabet {
	public void display(){ System.out.print("D");}
}
public class E extends Alphabet {
	public void display(){ System.out.print("E");}
}
public class Soup {
	private Alphabet[] bowl;
	public void go(){
		bowl = new Alphabet [5];//
		bowl [0] = new E();
		bowl [1] = new C();
		bowl [2] = new D();
		bowl [3] = new B();
		bowl [4] = new A();
		for (int i=0; i< bowl.length; i++)
			bowl [i].display();//Question 7
		System.out.println("");//move to new line
		change(bowl [0], new A());
		bowl [0].display();//Question 8
		System.out.println("");//move to new line
		bowl [0] = new E();
		change(bowl, new A(), 0);
		bowl [0].display();//Question 9
	}
	public void change(Alphabet a, Alphabet b){
		a = b;
	}
	public void change(Alphabet[] a, Alphabet b, int i){
		a[i] = b;
	}
	public static void main(String[] args){
		Soup x = new Soup();
		x.go();
	}//end main
}//end Soup
```

```
public class Wrapping {
  private int i;
```

```
  public Wrapping(int x) { i = x; }
  public int value() { return i; }
}

public class Parcel {
  public Wrapping wrapping(int x) {
          return new Wrapping(x){
                  public Wrapping(int x){
                          super(x);
                  }
                  public int value (){ return 47*i;}
          };
    //return an anonymous inner class object of Wrapping type
   //with overloaded method "public int value (){ return 47*i;}"
  }
  public static void main(String[] args) {
    Parcel p = new Parcel();
    Wrapping w = p.wrapping(10);
  }
}
```

In the above classes provide the missing code.

```
class Parcel4 {
  private class PContents implements Contents {
    private int i = 11;
    public int value() { return i; }
  }
  protected class PDestination implements Destination {
    private String label;
    private PDestination(String whereTo) {
      label = whereTo;
    }
    public String readLabel() { return label; }
  }
  public Destination destination(String s) {
    return new PDestination(s);
  }
  public Contents contents() {
    return new PContents(); }}
public class TestParcel {
  public static void main(String[] args) {
    Parcel4 p = new Parcel4();
    Contents c = p.contents();
    Destination d = p.destination("Tasmania");
   Parcel4.PContents pc = p.new PContents();//ERROR
  }
}
```

The above class has an error on the line indicated. Explain why there is an error.

```
public class MyExceptionTest {
	public void foo() throws Exception {
		System.out.println("foo");
	}
	public void bar() throws Exception {
		System.out.println("bar");
		throw new Exception();
	}
	public void test() throws Exception {
		System.out.println("starting test");
		try{
			foo();
			bar();
			System.out.println("finished try");
		}catch(Exception e){
			System.out.println("caught an exception");
		////////	throw e;
		}
		finally{
			System.out.println("finally");
		}
		System.out.println("finished test");
	}
	public static void main(String[] args) throws Exception{
		(new MyExceptionTest()).test();
	}
}
```

1. Give the output for the above code. Note the line commented out.

2. Give the output for the above code if the “throw e” line was not commented out

```
class Example{
	public void open() throws FileNotFoundException{
		System.out.println("attempting to open file");
		throw new FileNotFoundException();
	}
	public void close() throws CloseException {
		System.out.println("attempting to close file");
		throw new CloseException();
	}
	public static void main(String[] args) throws Exception{
		Example e = new Example();
		try{
			e.open();
			System.out.println("after opening file");
		}finally{
			System.out.println("finally");
			e.close();
			System.out.println("after closing file");
		}
		System.out.println("end of program");
    }
}
```

3. Give the output. State any exception(s) that are displayed on exit.

```
class LanguageException extends Exception{}
class JavaException extends LanguageException{}
public class Test {
	public void a() throws LanguageException{
		throw new LanguageException();
	}
	public void b() throws JavaException{
		throw new JavaException();
	}
	public static void main(String[] args){
		Test t = new Test();
		try{
			t.a();
			t.b();
		}
		catch(LanguageException l){}
		catch(JavaException j){}
		System.out.println("finished main");
```

```
        }
}
```

4. Give the output or indicate any errors and explain why.