The following problems cover material from all psets up to #3 which are what encompass this exam. This is to be used to test your knowledge and find gaps.

Inheritance, Static Members

1. Inheritance and Object Creation:

Given the class structure:

```java
public class A {
   public A() {}
   public A(int a, int b) {}
}

public class B extends A {
   public B(int r) {}
   public B(int r, int w) {
      super(r, w);
   }
}
```

Determine the legitimacy of the following constructor calls:

```
B c = new B();
A s = new B(1);
B c = new B(1, 9);
A t = new B(1, 9, 4);
B t = (new B(1)).new B(1);
B b = new A(1, 2);
```

For invalid calls, why are they invalid.

---

2. Dynamic Binding, Method Overriding and `toString`:

Given the classes:

```java
public class B {
   private int x;
   public int getX() {
      return x;
   }
   public String toString() {
      return x + "";
   }
}

public class E extends B {
   public int y=3;
   public String toString() {
      return getX() + y + "";
   }
}
```

What is the output of the following code?

```java
B b = new E();
System.out.println(b);
```

2b. Dynamic Binding in Polymorphism:

Observe the following code:

```java
class Animal {
    void makeSound() {
        System.out.println("Some sound...");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Bark");
    }

    void fetch() {
        System.out.println("Fetch the ball");
    }
}

class Cat extends Animal {
    void makeSound() {
        System.out.println("Meow");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        Animal myCat = new Cat();

        myDog.makeSound();
        myCat.makeSound();

        // Uncomment the following line
        // myDog.fetch();
    }
}
```

1. What is the output when the program is run?
2. Why does the uncommented call to `myDog.fetch()` result in a compile-time error even though the `myDog` object is an instance of the `Dog` class?
3. How can you modify the code to successfully call the `fetch` method on the `myDog` object?

2c. Dynamic Binding with Inheritance:

Imagine there are two classes: `Shape` and a subclass `ColoredShape` :

```java
public class Shape {
    void draw() {
        System.out.println("Drawing a shape.");
    }

    String getType() {
        return "Generic shape";
    }
}
```

```java
public class ColoredShape extends Shape {
    String color;

    ColoredShape(String color) {
        this.color = color;
    }

    @Override
    void draw() {
        System.out.println("Drawing a " + color + " shape.");
    }

    String getColor() {
        return this.color;
    }
}
```

Now, in the main method:

```java
public class ShapeApp {
    public static void main(String[] args) {
        Shape s1 = new Shape();
        Shape s2 = new ColoredShape("red");

        s1.draw();
        s2.draw();

        System.out.println("s1 is a: " + s1.getType());
        // Uncomment the following line
        // System.out.println("s2 has color: " + s2.getColor());
    }
}
```

1. What will be the output when you run the `ShapeApp` class?
2. The line `System.out.println("s2 has color: " + s2.getColor());` is commented out. Predict what would happen if you uncomment it and try to compile the program. Why does this occur?
3. How can you modify the code to access the `getColor()` method of `s2` object without causing a compilation error?

3. Static Methods and Inheritance:

Given:

```
public class V {
   public static int stuff() {
      return 1;
   }
}

public class W extends V {
   public static int stuff() {
      return 2;
   }
}
```

What is the output of:

```
V v = new W();
System.out.println(v.stuff());
```

4. Method Overloading and Inheritance:

```
class GrandParent { }
class Parent extends GrandParent{ }
class Child extends Parent { }

class Foo {
   public void bar(GrandParent p) {
      System.out.println("called with type GrandParent");
   }
   public void bar(Parent p) {
      System.out.println("called with type Parent");
   }
}
```

```
public class Test {
   public static void main(String[] args) {
      new Foo().bar(new Child());
   }
}
```

What is the output of this code? Explain.

---

5. Final Methods, Static Methods, and Inheritance (also requiring the creation of a basic method):

Given:

```
public class Parent {
   public final void printName() {
      System.out.println("I am in class Parent, dynamic invocation.");
   }
}

public class Child extends Parent {
   public void displayInfo() {
      printName();
      printClassName();
   }
   public static void printClassName() {
      System.out.println("I am in class Child, static invocation.");
   }
}
```

Write the output of the following code:

```
Parent p = new Child();
p.displayInfo();
```