Midterm Java

Question 1.

How do you invoke this program from the command line and make it print"foobar"?

```
#Java
//Question
public class Printer {
    public static void main(String[] args) {
        String parameterValue = System.getProperty("param");
        System.out.printf("%s%n", parameterValue);
    }
}

//Answer:
$: java -Dparam="foobar" Printer
```

Question 2.

Will this program print true when invoked?

```
//Question
public static void main(String[] args) {
        int seed = 300;
        byte b = (byte) seed;
        System.out.printf("%s%n", (b == seed));
    }

//Answer
No, this program will not print true when invoked, since 300 is larger than max byte
value with is 127.
So the largest value I can assign to seed to make this program print true is 127.
```

Question 3.

Will this program print when invoked?

```
//Question
public static void main(String[] args) {
        int a = -1, b = -1, c = 3;
        c += b *= a;
        System.out.printf("%d%n", c);
    }
}
//Answer
First, calculate b *= a, final b = 1
Second, calculate c += b, final c = 4
so, the program will print 4.
```

Question 4.

What will this program print when invoked?

```
//Question
public class Shader {

    public static void main(String[] args) {
        Shader shader = new Shader();
        int foo = shader.shade();

        System.out.printf("%d%n", foo);
    }

    private final int foo;

    public Shader() {
        this.foo = 10;
    }

    public int shade() {
        int foo = 0;
        for (int i = 0; i < 100; i++) {
            foo++;
        }

        return this.foo;
    }
}

//Anser
this program will print 10 when invoked, since the instance variable foo value has
never been changed.
```

Question 5.

When invoked will this program print 9, 10, 11 or it not complie?

```
//Question
public class Overloading {
    public static void main(String[] args) {
        Overloading overloading = new Overloading(10);

        int adjusted = overloading.getValue(1);
        System.out.printf("%d%n", adjusted);
    }

    private final int value;

    public Overloading(int value) {
        this.value = value;
    }

    public int getValue() {
        return this.value;
    }

    public int getValue(int increment) {
        return (value + increment);
    }

    public int getValue(int decrement) {
        return (value - decrement);
    }
}

//Anser
//1. it will not complie, since this class has two same getValue methods.
//2. to make this class more readable, and satisfy question's requirement, I will
change the method signature at line 19 to getIncrementValue(int increment)
//3. In my point of view, I think change the two method signatures to
getIncrementValue(int increment) and get DecrementValue(int decrement) can make this
class more readable.
```

Question 6.

This code does not complie, Fix it by changing the code to make it complie

```java
//Question
public class NotCompiling {
    private final int value;

    public int getValue(int decrement) {
        return (value - decrement);
    }

    public int getValue(boolean increment, int value) {
        return (increment ? (this.value + value) : (this.value) - value);
    }
}

//Answer
public class NotCompiling {
    private final int value;

    public int getValue(int decrement) {
        return (value - decrement);
    }

    public int getValue(boolean increment, int value) {
        return (increment ? (this.value + value) : (this.value) - value);
    }

    public NotCompiling(int value) {
        this.value = value;
    }
}
```

Question 7.

This code does not compile. Fix it by changing the code to make it compile.

```
//Question
public class NotCompiling {
    private int value;

    public int getValue(int decrement) {
        int value;
        return (value - decrement);
    }

    public int getValue(boolean increment) {
        int value;
        return (increment ? (value + 1) : (value) - 1);
    }
}

//Answer
public class NotCompiling {
    private int value;

    public int getValue(int decrement) {
        int value = 0;
        return (value - decrement);
    }

    public int getValue(boolean increment) {
        int value = 0;
        return (increment ? (value + 1) : (value) - 1);
    }
}
```

Question 8.

For these two classes, Parent and Child, which class does not compile?

```
//Question
public class Parent {
    private final String value;

    protected  Parent(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}

public class Child extends Parent {
    private final String something;

    public Child(String something) {
        this.something = something;
    }

    public String getSomething() {
        return something;
    }
}

//Answer
1.Child class does not compile
2.
public class Child extends Parent {
    private final String something;

    public Child(String value, String something) {
        super(value);
        this.something = something;
    }

    public String getSomething() {
        return something;
    }
}
```

Question 9.

For the following class hierarchy, 1) what are the super classes? 2) what are the subclasses?

```
//Question
public class Animal {
    private final String name;

    public Animal(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

public class Tiger extends Animal {
    public Tiger(String name) {
        super(name);
    }
}

public class Wolf extends Animal {
    public Wolf(String name) {
        super(name);
    }
}

//Answer
1)Animal is the super class of Tiger and Wolf, Object is the super class of Animal
2) Animal is the subclass of Object, Tiger and Wolf is the subclasses of Animal
```

Question 10.

Override getName in Tiger and overload getName in Wolf

```java
public class Tiger extends Animal {
    public Tiger(String name) {
        super(name);
    }
}

public class Wolf extends Animal {
    public Wolf(String name) {
        super(name);
    }
}

//Answer

//Override
public class Tiger extends Animal {
    public Tiger(String name) {
        super(name);
    }

    @Override public String getName() {
        return "Tiger's name is " + super.getName();
    }
}

//Overload
public class Wolf extends Animal {
    public Wolf(String name) {
        super(name);
    }

    public String getName(String greeeting) {
        return greeeting + " " + super.getName();
    }
}
```

Question 11.

Write your own subclass of Tiger called Bengal. Override getName and also add a property called location and a getter method called getLocation.

```
//Answer
public class Bengal extends Tiger {

    private final String location;

    public Bengal(String name, String location) {
        super(name);

        this.location = location;
    }

    @Override public String getName() {
        return "Bengal name is " + super.getName();
    }

    public String getLocation() {
        return this.location;
    }
}
```

Question 12.

[Unreadable]

```
//Question
public static void main(String[] args) {
        Bengal[] bengals = new Bengal[](new Bengal(), new Bengal());
        bengals[0] = new Bengal("A", "B");
        bengals[1] = new Bengal("C", "D");
        Animal[] animals = bengals;
        Wolf[] wolves = (Wolf[]) animals;
        bengals[1] = new Bengal("E", "F");
        System.out.printf("Hello %s%n", bengals[1]);
        wolves[1] = new Wolf("Grey thorn");
        System.out.printf("Hello %s%n", wolves[1]);
    }

//Answer
1. Class Bengal doesn't have a no-argument constructor
2. there way of declaring variable bengals is wrong, the correct way is:
    Bengal[] bengals = {new Bengal("A", "B"), new Bengal("C", "D")};
3. There will be ClassCastException since Wolf is not the super class of Bengal
```

Question 13.

Implement the equals and hashCode methods for the following subclass of Animal, Bear you cannot import any other class(es)

```
public class Bear extends Animal {
    private final String forest;

    private final int salmons;

    public Bear(String name, String forest, int salmons) {
        super(name);

        this.forest = forest;
        this.salmons = salmons;
    }

    //implement equals - use name, forest and salmons

    @Override public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj == this) return true;

        if (this.getClass() != obj.getClass()) return false;

        Bear bear = (Bear) obj;

        if (this.salmons != bear.salmons) return false;
        if (this.getName() != null ? !this.getName().equals(bear.getName()) :
bear.getName() != null) return false;
        if (this.forest != null ? !this.forest.equals(bear.forest) : bear.forest !=
null) return false;

        return true;
    }


    //implement hashCode
    @Override public int hashCode() {
        int forestResult = forest != null ? forest.hashCode() : 0;
        int nameResult = this.getName() != null ? this.getName().hashCode() : 0;
        return 31 * (forestResult + nameResult) + salmons;
    }
}
```

Question 14.

What happens when this program is executed? If it prints a value explicitly write which, If it fails explicitly write what exception is thrown.

```
//Question
public class Forest {
    public static void main(String[] args) {
        Forest forest = new Forest(100);
        forest.print(forest.getNumberOfTrees());
    }

    private final int numberOfTrees;

    public Forest(int numberOfTrees) {
        this.numberOfTrees = numberOfTrees;
    }

    public Integer getNumberOfTrees() {
        return (this.numberOfTrees > 10 ? null : this.numberOfTrees);
    }

    public void print(int numberOfTrees) {
        System.out.printf("Forest has %d trees%n", numberOfTrees);
    }
}

//Answer
this program will fail to execute, and it will throw a NullPointerException
```

Question 15.

Write the following two methods, One is a method named print which takes two arguments one is an Integer and the second is a var-args String, The second method is a main method which passes in the value 5 and the folowing String objects: “foo”, “bar”, “something”, The print method should print the number then do another print statement and print each of the passed in String values, on per line.

```
public class Solution {
    //first method
    public static void print(Integer num, String... args) {
        System.out.println(num);

        for (String s : args) {
            System.out.println(s);
        }
    }

    //second method, main method
    public static void main(String[] args) {
        print(5, "foo", "bar", "something");
    }
}
```

Question 16.

Implement the following Callback interface as an annoymous inner class and have it print the value foo, Change lines 4-6 to make WebServer print foo.

```
//Question
public interface Callback {
    void invoke();
}

public class WebServer {
    public static void main(String[] args) {
        int foo = 100;
        WebServer server = new WebServer();
        server.process();
    }

    public void process(Callback callback) {
        //do something

        try {
            Thread.sleep(1000L);
        } catch(InterruptedException ie) {
            Thread.currentThread().interrupt();
            throw new RuntimeException(ie);
        }
    }
}

//Answer
public static void main(String[] args) {
        final int foo = 100;

        Callback callback = new Callback() {
            @Override public void invoke() {
                System.out.println(foo);
            }
        };

        WebServer server = new WebServer();
        server.process(callback);
}
```

Question 17.

The following code is written using Java 1.7 try-with-resources. Rewrite the code so that it does not use try-with-resources and thus can be complied in Java 1.6.

```
//Question
public class Throwables {
    public static void main(String[] args) {
        String fileName = "/tmp/foo";
        try(InputStream stream = new FileInputStream(fileName)) {
            int read;

            while((read = stream.read()) != -1) {
                System.out.printf("%d%n", read);
            }
        } catch (IOException ioe) {
            System.out.printf("Error! %s", ioe.getMessage());
        }
    }
}

//Answer
public class Throwables {
    public static void main(String[] args) {
        String fileName = "/tmp/foo";
        try{
            InputStream stream = new FileInputStream(fileName);
            int read;

            while((read = stream.read()) != -1) {
                System.out.printf("%d%n", read);
            }
        } catch (IOException ioe) {
            System.out.printf("Error! %s", ioe.getMessage());
        }
    }
}
```

Question 18.

Create the following class hierarchy. There should be an interface named Animal which has two methods; getName which return a String and a getNumberLegs which returns an Integer, There should be an AbstractAnimal implementation of Animal that is an abstract class. it should implement the getName method but nothing else, Then create two subclasses of AbstractAnimal of any type of animal that you want.

```
//Answer

//Interface
public interface Animal {
    String getrName();
    Integer getNumberLegs();
}

//Abstract class
public abstract class AbstractAnimal implements Animal {
    private final String name;

    protected AbstractAnimal(String name) {
        this.name = name;
    }

    @Override public String getName() {
        return this.name;
    }
}

//Dog

public class Dog extends AbstractAnimal {
    public Dog(String name) {
        super(name);
    }

    @Override public Integer getNumberLegs() {
        return 4;
    }
}

//Bird
public class Bird extends AbstractAnimal {
    public Bird(String name) {
        super(name);
    }

    @Override public Integer getNumberLegs() {
        return 2;
    }
}
```