Java Basics & OOPs Assignment - Solutions

1. Java Basics

2. What is Java? Explain its features.

Java is a high-level, object-oriented programming language.

Features:

* Platform Independent: Write once, run anywhere (via JVM)
* Object-Oriented
* Robust
* Secure
* Portable
* Multithreaded
* Distributed

2. Explain the Java program execution process.

3. Write source code (.java file)

4. Compile using javac to generate bytecode (.class file)

5. JVM loads and executes bytecode on any platform

6. Write a simple Java program to display 'Hello World'.

```
public class HelloWorld {
public static void main(String\[] args) {
System.out.println("Hello World");
}
}
```

4. What are data types in Java? List and explain them.

Primitive types:
byte, short, int, long: Integer types
float, double: Floating point types
char: Character type
boolean: true/false

Non-primitive types:
Arrays
Classes
Interfaces

5. What is the difference between JDK, JRE, and JVM?

JVM: Runs Java bytecode
JRE: JVM + libraries to run Java applications
JDK: JRE + compiler + development tools

6. What are variables in Java? Explain with examples.

Variables store data values.

int age = 20;
float price = 99.99f;
String name = "Sakshi";

7. What are the different types of operators in Java?

Arithmetic: + - * / %
Relational: == != > < >= <=
Logical: && || !
Assignment: = += -= *=
Unary: ++ --
Bitwise: & | ^ << >> \~

8. Explain control statements in Java (if, if-else, switch).

if: Runs a block if condition is true

if-else: Runs one block if condition is true, else runs another block
switch: Matches a value with cases

9. Write a Java program to find whether a number is even or odd.

```
import java.util.Scanner;

public class EvenOdd {
public static void main(String\[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter number: ");
int num = sc.nextInt();
if (num % 2 == 0) {
System.out.println("Even");
} else {
System.out.println("Odd");
}
}
}
```

10. What is the difference between while and do-while loop?

while: Checks condition first, may not run at all
do-while: Executes once before condition is checked, always runs at least once

2. Object-Oriented Programming (OOPs)

3. What are the main principles of OOPs in Java?

Encapsulation: Binding data and methods together
Inheritance: Reusing code from a base class
Polymorphism: One name, many forms
Abstraction: Hiding internal details

2. What is a class and an object in Java? Give examples.

```
class Car {
String color;
void drive() {
System.out.println("Driving");
}
}
```

```
public class Main {
public static void main(String\[] args) {
Car c = new Car();
c.color = "Red";
c.drive();
}
}
```

3. Write a program using class and object to calculate area of a rectangle.

```
class Rectangle {
int length, breadth;
int area() {
return length \* breadth;
}
}


public class Main {
public static void main(String\[] args) {
Rectangle r = new Rectangle();
r.length = 10;
r.breadth = 5;
System.out.println("Area: " + r.area());
}
}
```

4. Explain inheritance with real-life example and Java code.

```
class Animal {
void sound() {
System.out.println("Animal makes sound");
}
}


class Dog extends Animal {
void bark() {
System.out.println("Dog barks");
}
}
```

```
public class Main {
public static void main(String\[] args) {
Dog d = new Dog();
d.sound();
d.bark();
}
}
```

5. What is polymorphism? Explain with compile-time and runtime examples.

Compile-time polymorphism (method overloading)

```
class Calc {
int add(int a, int b) {
return a + b;
}
int add(int a, int b, int c) {
return a + b + c;
}
}
```

Runtime polymorphism (method overriding)

```
class Parent {
void show() {
System.out.println("Parent show");
}
```

```
}


class Child extends Parent {
void show() {
System.out.println("Child show");
}
}
```

6. What is method overloading and method overriding? Show with examples.

Method overloading

```
class Demo {
void show(int a) {}
void show(int a, int b) {}
}
```

Method overriding

```
class A {
void display() {
System.out.println("A");
}
}


class B extends A {
```

```
void display() {
System.out.println("B");
}
}
```

7. What is encapsulation? Write a program demonstrating encapsulation.

```
class Student {
private int age;
public void setAge(int age) {
this.age = age;
}
public int getAge() {
return age;
}
}
```

```
public class Main {
public static void main(String\[] args) {
Student s = new Student();
s.setAge(20);
System.out.println(s.getAge());
}
}
```

8. What is abstraction in Java? How is it achieved?

Abstraction means hiding internal details and showing only essential features.

It is achieved using abstract classes and interfaces.

9. Explain the difference between abstract class and interface.

Abstract class:
Can have method bodies
Can have constructors
Supports single inheritance

Interface:
Only method signatures (Java 7)
No constructors
Supports multiple inheritance

10. Create a Java program to demonstrate the use of interface.

```
interface Animal {
void sound();
}

class Dog implements Animal {
public void sound() {
System.out.println("Dog barks");
}
}
```

```
public class Main {
public static void main(String\[] args) {
Dog d = new Dog();
d.sound();
}
}
```