|  |  |  |
| --- | --- | --- |
| Sr No | Code | Milli sec |
| 1 | for (long i = 0; i<1000000;i++) {   System.*out*.println("Hello " + i);  } | **3194** |
|  | for (long i = 0; i<1000000;i+=3) {   System.*out*.println("Hello " + i);  System.*out*.println("Hello " + i);  System.*out*.println("Hello " + i);  } | **2839** |
| 2 | List<Integer> list = new LinkedList<>();  for (int i = 1 ;i<1000000 ;++i ){  list.add(i); }  int size = list.size();  for ( int j = 0 ; j<list.size() ;++j){  System.*out*.println("Counter "+j); } | **1790** |
|  | for ( int j = 0 ; j<size ;++j){  System.*out*.println("Counter "+j); } | **1704** |
| 3. **Use of Conditional (Ternary) Operator** | for(int i=0 ; i<1000000 ; ++i){  isMember = i%2==1 ?true :false;  if (isMember) {  discount = 10;  } else {  discount = 5;  }   System.*out*.println("discount :"+discount); } | **1217** |
|  | for(int i=0 ; i<1000000 ; ++i){  isMember = i%2==1 ?true :false;   discount = isMember ? 10 : 5;   System.*out*.println("discount :"+discount); } | **1215** |
| 4 | for(int i = 0; i<100000 ;++i){  day = i%4;  if (day == 4) {  System.*out*.println("Sunday");  } else if (day == 2) {  System.*out*.println("Monday");  } else if (day == 3) {  System.*out*.println("Tuesday");  } else {  System.*out*.println("Invalid day");  } } | **226** |
|  | int day = 4  for (int i = 0; i < 100000; ++i) {  day = i%4;  switch (day) {  case 1:  System.*out*.println("Sunday");  break;  case 2:  System.*out*.println("Monday");  break;  case 3:  System.*out*.println("Tuesday");  break;  default:  System.*out*.println("Invalid day");  break;  } } | **202** |
|  | for (int i = 0; i < 100000; ++i) {  System.*out*.println( switch (day) {  case 1 -> "Sunday";  case 2 ->"Monday";  case 3 ->"Tuesday";  default->"Invalid day";  }); | **147** |
| 5. Use StringBuilder for string concatenation | String result = ""; for (int i = 0; i < 10000; i++) {  result += i;   System.*out*.println("result"+result);  } | **1096** |
|  | StringBuilder resultBuilder = new StringBuilder(); for (int i = 0; i < 10000; i++) {  resultBuilder.append(i);  System.*out*.println(resultBuilder); } | **1048** |
| 6. **Optimize I/O Operations** | try (FileReader reader = new FileReader("file.txt")) {  int character;  while ((character = reader.read()) != -1) {  System.*out*.print((char) character);  }  } catch (IOException e) {  e.printStackTrace(); | **606** |
|  | try (BufferedReader reader = new BufferedReader(new FileReader("file.txt"))) {  String line;  while ((line = reader.readLine()) != null) {  System.*out*.println(line);  } } catch (IOException e) {  e.printStackTrace(); } | **41** |
| 7. **Use of Logical AND (&&) and OR (||) Operators:** | for (int i = 0; i<100000 ; ++i){  if (x != 0 && y / x > 10) {  System.*out*.println("Without short-circuiting");  } } | **222** |
|  | for (int i = 0; i<100000 ; ++i){  if (x != 0 & y / x > 10) {  System.*out*.println("With short-circuiting");  } } | **192** |
| 8. String equal and length function comparison | for(int i = 0 ; i<1000000 ;++i){ if(s.equals(""))  System.*out*.println("Empty string"); } | **2462** |
|  | for(int i = 0 ; i<1000000 ;++i){  if(s.length()==0)  System.*out*.println("Empty string"); } | **1474** |
| 9. Optimize Algorithm Complexity | // Bad: Linear search  for (int i = 0; i < array.length; i++) {  if (array[i] == target) {  return i;  }  } |  |
|  | // Good: Binary search  Arrays.sort(array);  int index = Arrays.binarySearch(array, target); |  |
| 10. : Fixing the Memory Leak | public class SomeClass {  private staticList<String> list = new ArrayList<>();  public void addToList(String item) {  list.add(item);  // Ensure the list doesn't grow indefinitely  if (list.size() > 100) {  list.remove(0);  } } |  |
| 11. **Exceptions** | Int numTestLoops = 100,000;  public ArrayList testSystemException() {  ArrayList al = new ArrayList<>();  for (int i = 0; i < numTestLoops; i++){  Object o = null;  if ((i % exceptionFactor) != 0) {  o = new Object();   }   try {   al.add(o.toString());  } catch (NullPointerException npe) {  // Continue to get next string  }  }  return al;  } | 381ms |
|  | public ArrayList testCodeException() {  ArrayList al = new ArrayList<>();  for (int i = 0; i < numTestLoops; i++) {  try {  if ((i % exceptionFactor) == 0) {  throw new NullPointerException("Force Exception");  }  Object o = new Object(); al.add(o.toString());  } catch (NullPointerException npe) {  // Continue to get next string }  }  return al;  } | 15ms |
|  | public ArrayList testDefensiveProgramming() {  ArrayList al = new ArrayList<>();  for (int i = 0; i < numTestLoops; i++) {  Object o = null;  if ((i % exceptionFactor) != 0) {  o = new Object();  }  if (o != null) {  al.add(o.toString());  }    } return al;  } | 2 ms |
| **12. Choosing a Parser**  **XML StAX parser** | Running 20 iteration  XMLStreamReader reader = staxFactory.createXMLStreamReader(ins);  while (reader.hasNext()) {  reader.next();  int state = reader.getEventType();  switch (state) {  case XMLStreamConstants.START\_ELEMENT: String s = reader.getLocalName();  if (ITEM\_ID.equals(s)) {  isItemID = true;  }  break;  case XMLStreamConstants.CHARACTERS:  if (isItemID) {  String id = reader.getText();  isItemID = false;  if (addItemId(id)) { return;  }  }  break;  default: break;  } //end switch  } | 265ms |
| **JSON parser** | while (parser.hasNext()) {  Event event = parser.next();  switch (event) {  case KEY\_NAME:  String s = parser.getString();  if (ITEM\_ID.equals(s)) {  isItemID = true;  }  break;  case VALUE\_STRING:  if (isItemID) {  if (addItemId(parser.getString())) {  return;  }  isItemID = false;  }  continue;  default:  continue;  }  } | 146ms |
| **Push parsers (SAX)** | protected class CustomizedInnerHandler extends DefaultHandler {  public void startElement(String space, String name, String raw, Attributes atts) {  if (name.length() == 0) name = raw;  if (name.equalsIgnoreCase(ITEM\_ID))  isItemID = true;  }  public void characters(char[] ch, int start, int length) throws SAXDoneException {  if (isItemID) {  String s = new String(ch, start, length);  isItemID = false;  if (addItemId(s)) {  throw new SAXDoneException("Done");  }  }  }  } | 231 ms |
|  |  |  |
|  |  |  |